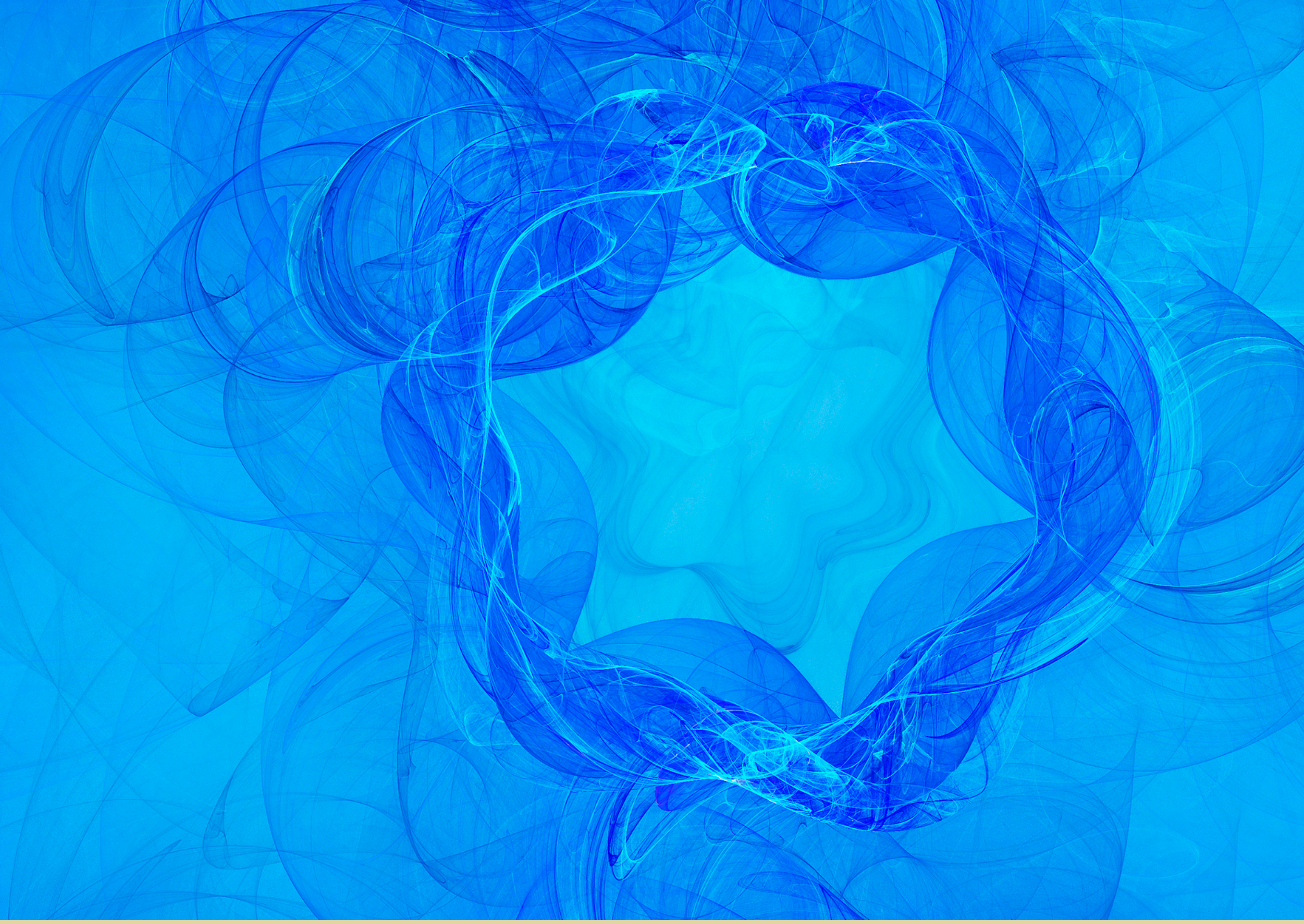


Ivo Carlos Pereira Gonçalves

An Exploration of Generalization and Overfitting in Genetic Programming: Standard and Geometric Semantic Approaches

Doctoral thesis submitted to the Doctoral Program in Information Science and Technology, supervised by Associate Professor Carlos Manuel Mira da Fonseca and Principal Investigator Sara Guilherme Oliveira da Silva, and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

November 2016

· U C ·

Universidade de Coimbra

An Exploration of Generalization and Overfitting in Genetic Programming: Standard and Geometric Semantic Approaches

A thesis submitted to the University of Coimbra
in partial fulfillment of the requirements for the
Doctoral Program in Information Science and Technology

by

Ivo Carlos Pereira Gonçalves
icpg@dei.uc.pt

Department of Informatics Engineering
Faculty of Sciences and Technology
University of Coimbra

Coimbra, November 2016

Financial support by Fundação para a Ciência e a Tecnologia,
SFRH/BD/79964/2011.

An Exploration of Generalization and Overfitting in Genetic Programming:
Standard and Geometric Semantic Approaches

This dissertation was prepared under the supervision of

Carlos Manuel Mira da Fonseca
Associate Professor
of the Department of Informatics Engineering
of the Faculty of Sciences and Technology
of the University of Coimbra

and
Sara Guilherme Oliveira da Silva
Principal Investigator
of the Department of Informatics
of the Faculty of Sciences
of the University of Lisbon

Colorless green ideas sleep furiously
Noam Chomsky, Syntactic Structures (1957)

Acknowledgements

I would like to thank my main supervisor Sara Silva for all her support throughout this journey, and for giving me the liberty to explore my ideas. Thanks are also due to my other supervisor Carlos Fonseca for joining us and for always providing a much needed different perspective.

A general thanks to the Evolutionary and Complex Systems (ECOS) group. A special mention is due to Nuno Lourenço for all the scientific discussions and for all the moments shared outside of work. Thanks to my parents for their love and for their contributions to my education. Thanks to all of those that are not explicitly mentioned but that somehow contributed to this journey.

Last but not least, thanks to Joaninha for everything :)

Ivo Gonçalves
Coimbra, November 2016

Resumo

Aprendizagem computacional refere-se à tarefa de induzir um padrão general a partir de um conjunto de exemplos. Espera-se que um método de aprendizagem consiga generalizar para exemplos não vistos do mesmo padrão. Um problema comum em aprendizagem computacional é a possibilidade de os modelos resultantes aprenderem simplesmente o conjunto de exemplos dado, em vez de aprender o padrão subjacente. Quando um modelo tem este comportamento, diz-se que ele está em sobreajustamento. Esta dissertação explora a tarefa de aprendizagem computacional e os conceitos relacionados de generalização e sobreajustamento, no contexto da Programação Genética (PG). PG é um método computacional inspirado pela evolução natural que considera um conjunto de funções primitivas e de terminais, que podem ser combinados sem nenhuma restrição considerável em relação à estrutura dos modelos que estão a ser evoluídos. Esta flexibilidade pode ajudar a aprendizagem de padrões complexos mas também aumenta o risco do sobreajustamento.

As contribuições desta dissertação cobrem a forma mais comum de PG (Standard PG), assim como a recentemente proposta Programação Genética Geométrica Semântica (PGGS). O conjunto de abordagens iniciais é baseado numa seleção dinâmica de diferentes subconjuntos dos dados de treino durante o processo evolucionário. Estas abordagens conseguem evitar o sobreajustamento e melhorar a generalização resultante sem restringir a flexibilidade da PG. É realizada uma análise da capacidade de generalização da PGGS, que demonstra que a generalização resultante é consideravelmente dependente das características do operador de mutação. É demonstrado que, tal como Standard PG, a formulação original de PGGS tem tendência a sobreajustar. São apresentadas as condições necessárias para evitar o sobreajustamento. Quando essas condições se verificam, PGGS consegue alcançar uma generalização particularmente competitiva. É proposta uma nova mutação geométrica semântica que melhora de forma substancial a eficácia e a eficiência da PGGS. Além de melhorar consideravelmente o ritmo de aprendizagem dos dados de treino, também consegue alcançar uma generalização competitiva com apenas algumas aplicações do operador de mutação.

O conjunto final de contribuições cobre o domínio das Redes Neuronais (RNs). Estas contribuições resultaram de uma extensão da investigação realizada em PGGS. Este conjunto de contribuições inclui a definição de um algoritmo de construção de RNs baseado numa extensão do operador de mutação da PGGS. De forma semelhante à PGGS, o algoritmo proposto pesquisa sobre um espaço sem óptimos locais. Isto permite realizar uma pesquisa estocástica eficaz e eficiente no espaço das RNs, sem existir a necessidade de usar retropropagação para ajustar os pesos da rede. Finalmente, são propostos dois critérios de paragem de pesquisa que conseguem detectar quando o risco de sobreajustamento sobe consideravelmente. É mostrado que os pontos de paragem detectados resultam numa generalização competitiva.

Abstract

Computational learning refers to the task of inducing a general pattern from a provided set of examples. A learning method is expected to generalize to unseen examples of the same pattern. A common issue in computational learning is the possibility that the resulting models could be simply learning the provided set of examples, instead of learning the underlying pattern. A model that is incurring in such a behavior is commonly said to be overfitting. This dissertation explores the task of computational learning and the related concepts of generalization and overfitting, in the context of Genetic Programming (GP). GP is a computational method inspired by natural evolution that considers a set of primitive functions and terminals that can be combined without any considerable constraints on the structure of the models being evolved. This flexibility can help in learning complex patterns but it also increases the risk of overfitting.

The contributions of this dissertation cover the most common form of GP (Standard GP), as well as the recently proposed Geometric Semantic GP (GSGP). The initial set of approaches relies on dynamically selecting different training data subsets during the evolutionary process. These approaches can avoid overfitting and improve the resulting generalization without restricting the flexibility of GP. Besides improving the generalization, these approaches also produce considerably smaller individuals. An analysis of the generalization ability of GSGP is performed, which shows that the generalization outcome is greatly dependent on particular characteristics of the mutation operator. It is shown that, as Standard GP, the original formulation of GSGP is prone to overfitting. The necessary conditions to avoid overfitting are presented. When such conditions are in place, GSGP can achieve a particularly competitive generalization. A novel geometric semantic mutation that substantially improves the effectiveness and efficiency of GSGP is proposed. Besides considerably improving the training data learning rate, it also achieves a competitive generalization with only a few applications of the mutation operator.

The final set of contributions covers the domain of Neural Networks (NNs). These contributions originated as an extension of the research conducted within GSGP. This set of contributions includes the definition of a NN construction algorithm based on an extension of the mutation operator defined in GSGP. Similarly to GSGP, the proposed algorithm searches over a space without local optima. This allows for an effective and efficient stochastic search in the space of NNs, without the need to use backpropagation to adjust the weights of the network. Finally, two search stopping criteria are proposed, which can be directly used in the proposed NN construction algorithm and in GSGP. These stopping criteria are able to detect when the risk of overfitting increases significantly. It is shown that the stopping points detected result in a competitive generalization.

Keywords

Evolutionary Computation, Genetic Programming, Geometric Semantic Genetic Programming, Supervised Learning, Generalization, Overfitting, Neural Networks

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Within Computer Science, Artificial Intelligence (AI) is commonly seen as one of the most fascinating fields. A particularly important AI goal is the development of systems (or agents) that exhibit general intelligence. Usually, general intelligence is defined in terms of tasks that humans can perform that require a significant degree of intelligence. A system that is able to perform these tasks can be described as human-competitive. An important and related AI goal is the development of systems that can continuously improve without external guidance. The ability to learn and perform induction is crucial to achieve these important goals. In this context, learning refers to the task of inducing a general pattern from a provided set of examples. In other words, a learning method is expected to generalize to unseen examples of the same pattern. A common issue in computational learning is the possibility that the resulting models could be simply learning the provided set of examples, instead of learning the underlying pattern. A model that is incurring in such a behavior is commonly said to be overfitting. A learning method must avoid overfitting in order to be successful.

One of the areas that has been intensively studying learning is Genetic Programming (GP). GP is part of a class of computational methods inspired by natural evolution. One of the ultimate goals of GP is to evolve complex computer programs with as little human interaction as possible. Perhaps the most defining characteristic of GP is the flexibility of the models that can be evolved. Most learning methods set a priori constraints on the models or functions being considered. These constraints are usually expressed by considering only some particular class of functions, or by defining the structure of the function and only allowing some parameters to be searched on. The rationale behind these approaches is based on limiting the complexity of the resulting models or functions. Limiting this complexity might reduce the risk of overfitting, but it might also constraint the learning method to the point that the pattern can not be effectively learned. On the other hand, GP considers a set of primitive functions and terminals that can be combined without any considerable constraints on the structure of the models being evolved. In other words, GP allows the evolutionary search to implicitly define how complex the resulting individuals should be based on the chosen performance measure. The downside of this flexible approach is that the risk of overfitting might increase. For this reason it is particularly important to devise approaches that allow GP to learn without incurring in overfitting. Although the GP area has been studying learning tasks for a while now, a considerable amount of research has only considered the performance on the provided set of examples, instead of the generalization performance. Furthermore, although the interest in studying generalization and overfitting has been recently increasing in GP, it is still understudied in comparison with more well-established learning methods.

The focus of this dissertation is to study, within the context of GP, the task of learning and the related concepts of generalization and overfitting. The main objective is to develop approaches that can avoid overfitting and increase the resulting generalization, while maintaining the flexibility of GP. The proposed approaches should not constraint the complexity of the resulting models. Evolution should define how complex the models need to be given the particular task at hand. This dissertation studies the learning tasks where the expected outputs are real-valued (regression).

1.2 Contributions

The contributions of this dissertation cover three main domains: Standard GP, Geometric Semantic GP (GSGP), and Neural Networks (NN). The contributions on the last domain originated from an extension of the research being conducted in GSGP. The main contributions are the following:

- A set of approaches based on different usages of the available training data that can avoid overfitting in Standard GP. These approaches rely on dynamically selecting different training data subsets. Besides increasing generalization, the proposed approaches also result in considerably smaller individuals. This set of approaches achieves these outcomes without restricting the flexibility of GP, or imposing any particular limitation on the complexity of the evolved individuals. These outcomes are possible even in the extreme case of using a single training instance at each generation (Gonçalves et al., 2012). Within this set of approaches, some are aimed at balancing the overfitting avoidance component with an increased training data learning rate (Gonçalves and Silva, 2013). Besides the datasets considered in this dissertation, this set of approaches has also been applied in other real-world datasets (Silva et al., 2013).

- An analysis of the generalization ability of GSGP, showing that the generalization outcome is greatly dependent on particular characteristics of the mutation operator. It is shown that, as Standard GP, the original formulation of GSGP is prone to overfitting. The necessary conditions to avoid overfitting are presented. When such conditions are in place, GSGP can achieve a particularly competitive generalization. A connection with the Ensemble Learning area is also provided (Gonçalves et al., 2015a).

- A novel geometric semantic mutation that substantially improves the effectiveness and efficiency of GSGP. Besides considerably improving the training data learning rate, it also achieves a competitive generalization with only a few applications of the mutation operator (Gonçalves et al., 2015a).

- A different way of conducting the search that can learn the underlying patterns by effectively finding aligned individuals with arbitrary precision. A comparison with the traditional search approach is performed (Gonçalves et al., 2016).

- The definition of a NN construction algorithm based on an extension of the mutation operator defined in GSGP. Similarly to GSGP, the proposed algorithm searches over a space without local optima. This allows for an effective and efficient stochastic search in the space of NNs. The proposed construction algorithm excludes the need to use backpropagation to adjust the weights of the network. The mutation operator used within the proposed algorithm allows for a simple way of controlling the sparseness at each mutation operation. This results in a structural simplification of the resulting NN (Gonçalves et al., 2015b).

- The specification of two search stopping criteria that can be directly used in the proposed NN construction algorithm and in GSGP. These stopping criteria are able to detect when the risk of overfitting increases significantly. It is shown that the stopping points detected result in a competitive generalization, as well as smaller NNs or individuals.

1.3 Structure

The remainder of this document is organized as follows. Chapter 2 provides contextualization for this dissertation. Chapter 3 describes the experimental methodology used. Chapter 4 studies the effects of incorporating different training data sampling strategies within Standard GP. Chapter 5 explores the generalization ability of GSGP, and how different mutation implementations can considerably influence the generalization outcome. An alternative way of conducting search is also explored. Chapter 6 introduces a NN construction algorithm by extending the geometric semantic mutation to NNs. The proposed mutation operator is also used to select appropriate stopping points within the search process. Chapter 7 concludes and presents some future directions.

2 Background

This chapter provides the necessary contextualization for this dissertation. Section 2.1 introduces the task of supervised learning. Section 2.2 overviews Genetic Programming (GP). Section 2.3 reviews previous GP studies dealing with generalization and overfitting. Section 2.4 provides a discussion centered on the generalization issue in GP.

2.1 Supervised Learning

2.1.1 Overview

In supervised learning, the goal is to build a model that describes the underlying pattern of a set of input-output examples. In this context, a model might also be referred to as a function or as a hypothesis. Depending on the type of outputs of the examples, the learning task could be defined as a regression (real-valued outputs) or as a classification (discrete class labels). Binary classification is a special case of classification where there are only two possible class labels. The outputs of the examples are also known as targets. A crucial aspect of supervised learning is that the input-output examples provided to the learning algorithm (referred to as training data), only represent a part of the complete set of input-output examples of a given learning task. In real-world scenarios, the available data is commonly only a very small fraction of the complete set of input-output examples. However, the goal is not to build a model that explains the training data, but to build a model that explains the general pattern of the underlying learning task. In other words, the goal is to generalize from the training data. Because of this, supervised learning might also be referred to as inductive learning. A learning algorithm must then define a strategy to generate models with the best possible generalization ability. The generalization ability (or simply generalization) of a model is defined by its performance in data other than the training data. In practice, this generalization ability is estimated by leaving out of the training data a part of the total available data. The data left out of the training data is usually referred to as unseen data, testing data, or test data. A model that is performing well in unseen data is said to be generalizing. However, performance in training and unseen data does not always agree. Particularly, it is experimentally observable that some models perform well in training data, but poorly in unseen data. Such models are said to be overfitting the training data. This means that these models represent patterns that only occur in the training data. Another way of describing this scenario is to say that these particular models are memorizing the training data. Notice that it is trivial to create a model that completely memorizes the training data. This however is futile since the goal, as previously mentioned, is to find models that generalize from the training data. A model that is unable to properly fit the training data is said to be underfitting.

Figure 2.1 presents typical evolutions of the training and generalization performance (as measured by an error function) of an iterative supervised learning method (e.g., GP or an Artificial Neural Network). The line labeled as generalization optimum loosely represents the theoretically best possible outcome in terms of generalization. This does not mean that it is not possible to create a model from a given search space that perfectly models the data (in stochastic methods as GP this might be possible even by chance). It means that, with the amount of training data available, the probability of creating and choosing a perfect model is considerably low. In other words, a search method is not capable of consistently selecting models that perform better than this loosely defined generalization optimum. The exact value of such a generalization optimum is significantly influenced by the amount of training data available, as in general an induction becomes more likely to succeed as the available training data increases. Two possible scenarios are presented for the generalization error evolution. Initially and for both scenarios, the training and generalization errors present similar values and decrease in similar amounts. As the training error gets closer to the value of the generalization optimum, the risk of overfitting starts to increase. In the first generalization scenario, overfitting starts to greatly increase as the generalization optimum is passed. In the second scenario, the generalization starts to stabilize. However, even in this second scenario, the generalization error starts to slowly increase if the training error continues to converge to zero. These different outcomes result from different ways of formulating an inductive search.

From the extensive experimental results accumulated throughout the application of several inductive learning algorithms, it stands clear that overfitting is a phenomenon that must be taken into consideration when designing a supervised learning method. Particularly, it is clear that a method that sets the reduction of the training error to zero as the only criterion, is likely to overfit. One of the only scenarios where this strategy would be promising is that where the amount of training data available is rather plentiful, which could allow the construction of models with perfect generalization. However, in real-world scenarios this seldom occurs. The probability of overfitting also increases if no care is taken regarding the structure of the models that are being considered. Successful learning methods must allow for flexibility and complexity in the creation of the models in order to learn the underlying pattern, while at the same time defining strategies to avoid overfitting the training data. Besides the complexity of the models, other factors might influence the learning process such as the presence of noise and the representativity of the provided data. Some learning methods can be considerably sensitive to the presence of noise. In some domains, particular noise characteristics might be known a priori. When this is the case, the learning method might be changed to incorporate this information into the learning process. As in the classical GP approach, this dissertation does not explicitly address the issues of noise and data representativity.

2.1.2 Theoretical Considerations

The design of learning algorithms is usually influenced by theoretical considerations. Perhaps the most influential principle in the design of learning algorithms is Occam's razor. It is also considered one of the most important principles in modern science, being a commonly accepted general guiding rule in both theory and practice. Occam's razor states (Tornay, 1938):

> Entities should not be multiplied beyond necessity.

Another common formulation is:

> No more things should be presumed to exist than are absolutely necessary.

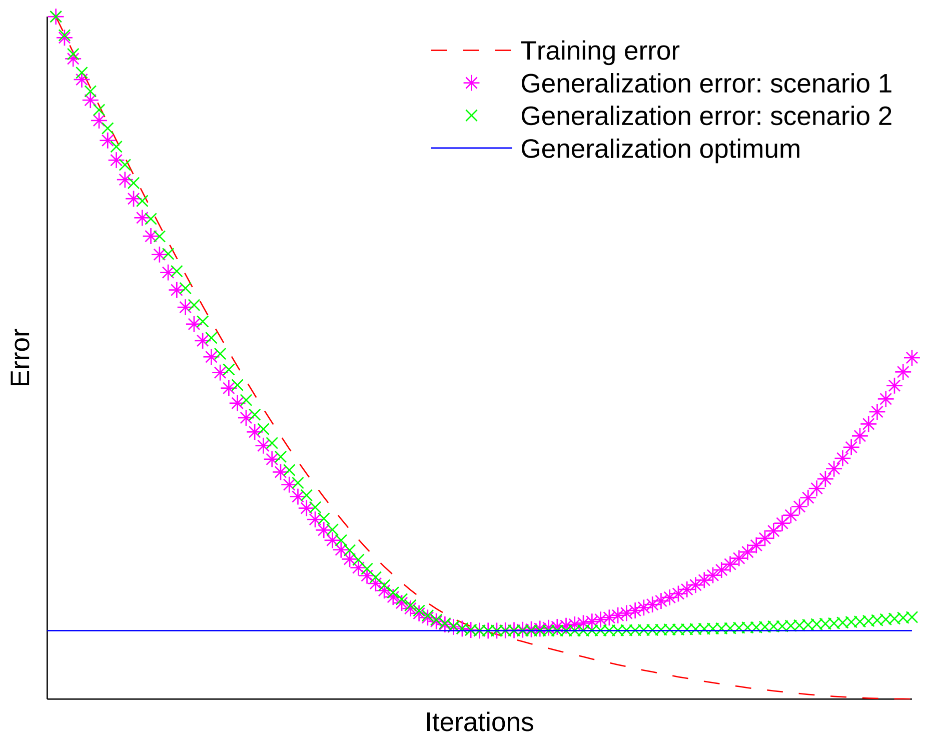


Figure 2.1: Typical evolutions of the training and generalization errors of an iterative supervised learning method across its iterations

One of the most famous formulations based on Occam's razor was stated by Isaac Newton:

> We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances.

Occam's razor entails a preference or bias toward simplicity, while allowing for added complexity when such is capable of explaining some phenomenon that could not be explained with lower complexity. Although this principle is not directly related to inductive learning, it is still a source of inspiration in the design of learning algorithms. Occam's razor is often interpreted in supervised learning as: simpler models generalize better. Alternatively, it is also interpreted as: given models with similar training performance, the simplest model should be preferred. However, these interpretations are not universally accepted (e.g., Domingos (1999)). Regardless, a considerable number of supervised learning methods are based on these interpretations. These particular methods incorporate a bias toward simplicity into their functioning. Another influential principle in the design of learning algorithms is the Minimum Description Length (MDL) (Rissanen, 1978). The MDL states:

> The best model is the one that minimizes the amount of information needed to encode it.

The supervised learning methods inspired by the MDL interpret it as: smaller models generalize better. Notice that this interpretation can be seen as a special case of the Occam's razor interpretation, as the size of a model can be considered a measure of its complexity. The learning methods inspired by the MDL principle incorporate a bias toward smaller models into their functioning.

Statistical Learning Theory (SLT) (Vapnik, 1995) is by now a mature field that provides theoretical considerations to guide the design of learning algorithms. STL includes two main induction principles: the Empirical Risk Minimization (ERM) and the Structural Risk Minimization (SRM). The ERM principle is intended for scenarios where the training data sample size is large. In these scenarios the training error (referred to as empirical risk) is close to the expected generalization error (referred to as expected risk), even if the training error is small. This means that overfitting is not a concern. However, these are mostly theoretical scenarios as in real-world scenarios the available training data is commonly only a small portion of the complete set of data. In these common real-world scenarios the ERM principle is unable to guarantee a small expected generalization error even if the training error is small. The SRM principle is formulated to address this issue. Central to the SRM principle is the notion of complexity or capacity of a set of functions/-models. This complexity may be expressed by the Vapnik-Chervonenkis (VC) dimension. The SRM principle implies that, besides reducing the training error, a successful learning algorithm must also constraint the VC dimension of the set of functions/models that are being considered by the algorithm. This principle suggests that to ensure the generalization of the resulting models, a tradeoff is needed between the reduction of the training error and the complexity of the functions/models. If the set of functions/models selected has a complexity (VC dimension) lower than what is need to learn the pattern, then underfitting will occur. On the other hand, if the selected complexity is too high, overfitting might occur. Consequently, the selection of a set of functions/models with an appropriate complexity for a given supervised learning task is a crucial component of the SRM principle. This focus on controlling the underlying complexity has a connection with the previously mentioned interpretations of the Occam's razor. Both Occam's razor and the SRM principle share the bias for less complexity.

Another influential set of ideas comes from Ensemble Learning (e.g., Hansen and Salamon (1990)). Ensemble Learning algorithms create several models for the same learning task, and combine them to produce a final model. This final model is known as an ensemble. The reasoning is that combining several models can reduce the overall risk of the final model being overfitted. This originates from the fact that the selection of a model always entails a risk of choosing an overfitted model, given that a priori there is no way of knowing which model will generalize or overfit. By forming an ensemble, even if some overfitted models are present, their negative contribution to the final model will be reduced since the final model will also include contributions from models which generalize well. This way of constructing the final model allows for a more robust generalization. This can be formally analyzed through the bias-variance decomposition. The bias-variance decomposition (e.g., Friedman (1997)) is a statistical concept that analyzes learning algorithms through the components of bias and variance. In the bias-variance decomposition, the analysis is conducted by considering several training data resamplings in a given learning task. This procedure results in a characterization of the behavior of a given learning algorithm in a given learning task as the training data varies. The bias component characterizes the average performance of the algorithm. The variance component characterizes how the performance fluctuates as the training data varies. In other words, the bias represents the quality of the approximation, and the variance represents the consistency or sensitivity of the approximation. Consequently, a learning algorithm that generalizes well in a given learning task, should present both low bias and low variance. An ensemble based approach can reduce the variance of the underlying learning algorithm (Breiman, 1996b). This contributes to a more robust generalization. The generalization achieved also tends to stabilize as more models are added to the ensemble. In comparison with the Occam's razor and the SRM principle, Ensemble Learning presents a considerably different perspective on how a learning algorithm should be designed. Ensemble Learning shows that large and potentially very complex models can indeed generalize well depending on how they are constructed.

The No Free Lunch (NFL) results (Wolpert, 1996, 2002) address the possibility of designing a superior general learning algorithm. A similar and related result is the law of conservation of generalization performance (Schaffer, 1994). The NFL results show that there are no a priori distinctions between learning algorithms. In other words, all learning algorithms perform the same, in terms of generalization, over all possible learning tasks. This is valid even when comparing against a random guessing algorithm. These results also imply that any learning algorithm that takes into account the complexity of the resulting models, performs the same as any learning algorithm that does not take complexity into account. Therefore, under the NFL results, the Occam's razor and the SRM principle are futile as mechanisms to guide the design of a superior general learning algorithm. This results from the fact that if they achieve a superior generalization in a given learning task, then they also assuringly result in an inferior generalization in another learning task. However, the NFL results do not address specific learning tasks. This implies that it is still relevant to study which learning algorithms perform better in each specific learning task. Also an important result is the fact that under some specific error functions, there are in fact a priori distinctions between learning algorithms, i.e., there are free lunches. Particularly, this scenario occurs if the error function induces a geometrical structure over the targets. This interestingly relates with Geometric Semantic GP (Moraglio et al., 2012), a recently proposed variant of GP which is formulated under a metric structure in the error function. This free lunch was already explored in GP (Poli et al., 2009) before the proposal of Geometric Semantic GP. Other free lunches were also explored within

GP (Poli and Graff, 2009).

2.1.3 Inductive Bias in Practice

From the above-mentioned theoretical considerations, different learning algorithms with different overfitting avoidance strategies emerge. The inductive bias of a learning algorithm can be defined as the a priori preference for models with particular characteristics. For instance, a learning algorithm that prefers small models, biases the search toward this objective. This can be done by restricting the search space to include only relatively small models, or it can be done by explicitly stating that preference in the error function that is applied to each model. This subsection overviews some of the most common inductive bias used in some of the most successful non-evolutionary supervised learning methods. This overview is by no means meant to be exhaustive. The different inductive bias applied in GP are explored in section 2.3.

One of the most common inductive bias found in practice is the simplicity bias, i.e., the preference for less complex models. This type of bias is related to the above-mentioned interpretations of the Occam's razor. In Decision Tree approaches such as Classification And Regression Trees (CART) (Breiman et al., 1984) and C4.5 (Quinlan, 1993), it is common to apply pruning (Quinlan, 1987). Pruning, as the name entails, is based on removing branches of a tree that have no significant contribution. Since this inductive bias leads to smaller and potentially less complex trees, the expected effect is the increased generalization of the resulting trees. Another common approach is the application of regularizations (e.g., Schölkopf and Smola (2002)). Loosely, a regularization is an approach that adds a component to the error function, in a way that is meant to penalize some unwanted characteristics of a model. This component is called a regularization term. Regularization is commonly used to penalize more complex models. For instance, a regularization term could be the size of a model, e.g., the size of a tree. An approach that wants to search for trees with good performance while at the same time biasing toward smaller trees, can do so by adding a regularization term that penalizes bigger trees. A commonly used regularization in Artificial Neural Networks is weight decay (e.g., Moody et al. (1995)). Weight decay works by limiting the growth of the weights of the network. It achieves this by adding a regularization term that penalizes bigger weights. This biases the search to accept bigger weights only when these can achieve considerably better performance than smaller weights. Weight decay is intended to decrease the overall complexity of the network. By doing so, it is expected that overfitting becomes less likely. It is common for already existing learning algorithms to be later updated to include regularization terms. This was not the case of Support Vector Machines (SVMs) (Boser et al., 1992). In SVMs the notion of controlling the underlying complexity was taken into account from the start of the development of the algorithm. This originates from the fact that SVMs are based on the SRM principle. As previously seen, this principle is based on indirectly controlling the complexity of the resulting functions/models by controlling the complexity of the set of functions/models. SVMs were formulated for binary classification but similar methods have been later proposed to regression. SVMs work by maximizing the margin between the decision boundary and a set of significant training instances. These significant training instances are now known as support vectors (originally called support patterns). In SVMs and their related approaches, the underlying complexity can be directly controlled by a specific parameter. Early stopping (e.g. Prechelt (1998)) is a common approach that is also connected with the simplicity bias. This approach can be applied to any iterative learning algorithm. The most common early stopping approach works by leaving out part of the training data to be used to estimate the generalization during the training phase. The part of the data used for this purpose is called the validation set. During the training the performance in the validation set is periodically computed. Typically, the error in the training and the validation set decreases up to a point where the error on the validation set starts to increase. This usually signifies that overfitting is starting to occur. Therefore, the training phase can be stopped. This potentially avoids incurring in higher overfitting. Early stopping can be seen as a type of indirect regularization. Since the training phase is stopped sooner, this usually results in smaller/less complex final models. According to the reasoning underlying the simplicity bias, this might translate into models with better generalization. Notice that under the conditions where the above-mentioned NFL results are applicable, all of these simplicity preferences and regularizations have no valid claim as being a better inductive bias than any other possible inductive bias, when it comes to all possible learning tasks.

The approaches based on Ensemble Learning do not have such an explicit inductive bias as the previous ones. They are mainly concerned with how the final model (ensemble) is constructed, and less with which inductive bias each individual model originates from. Notice that an ensemble can combine models from different learning algorithms and with different representations. Bagging (Breiman, 1996a) and Boosting (Freund and Schapire, 1995; Schapire, 1990) are two of the most influential Ensemble Learning methods. Bagging works by producing several different training sets. These different training sets are constructed by selecting training instances uniformly and with replacement from the original training set. A baseline learning algorithm is used to train with the different training sets. The final model is a combination of all the models produced with all the generated training sets. Boosting works in a similar fashion but maintains a set of weights over the original training set. It iteratively adjusts these weights by reducing the weights of the training instances which are correctly classified, and by increasing the weights of the training instances which are incorrectly classified. Given the flexibility of these approaches, the underlying inductive bias is perhaps better characterized by the nature of the baseline learning algorithm, i.e., the algorithm used to learn each different training set. Notice also that it is even possible to use different baseline learning algorithm for all the different training sets constructed.

2.2 Genetic Programming

Genetic Programming (GP) (Koza, 1992) encompasses a set of approaches aimed at evolving computer programs. By defining a set of program elements and a fitness function, GP is able to search the underlying space of computer programs. GP is the most recent major branch of the broader area of Evolutionary Computation (EC). The other three major branches are: Genetic Algorithms (Holland, 1975), Evolutionary Programming (Fogel et al., 1966), and Evolution Strategies (Beyer and Schwefel, 2002). GP was popularized by Koza (Koza, 1992), but earlier approaches aimed at evolving computer programs had already been explored (Cramer, 1985; Forsyth, 1981; Koza, 1989). The most common GP version uses a tree-based representation to store the evolved computer programs. This is known as Standard GP, and it is the version used in this dissertation. Other representation possibilities include, among others, storing the computer programs as sequential instructions of code (Brameier and Banzhaf, 2007) or as graphs (Miller and Thomson, 2000). The GP algorithm can be described by the following high-level steps:

1. Create a random initial population

2. Repeat the following operations until a given stopping criterion is met (usually a number of generations):
    2.1. Apply parent selection to select the individuals that are going to be used as parents
    2.2. Apply the variation operators to the parents to create a population of offspring
    2.3. Select the new population by applying survival selection to the previous population and the offspring
3. Return the best individual from the population according to fitness

An overview of the main GP components is conducted in the next subsections. Given that this dissertation is concerned with supervised learning, the following descriptions assume a supervised learning task.

2.2.1 Representation and Fitness Evaluation

GP uses a tree-based representation where the internal nodes are functions and the leaves are terminals. In this context, a function can refer to any particular computation with any number of arguments. A terminal can be an input variable of the learning task or a constant. In the uncommon case that, in supervised learning, the need would arise to include functions that do not take any argument, these functions would be represented as leaves in the tree. The functions are defined in a function set, and the terminals are defined in a terminal set. The combination of both of these sets is the primitive set. The primitive set contains all the elements that could be part of a GP individual. Figures 2.2 and 2.3 present examples of GP individuals. An important GP characteristic is that the available elements can be combined as freely as possible. In order for this to be possible, it must be guaranteed that all combinations of functions and arguments are valid. This is known as the closure property (Koza, 1992). To assure this property, type and evaluation safety must be achieved. Type safety can be achieved by having all functions and terminals defined as the same type, e.g., all real-valued. This is the approach taken in Standard GP. In alternative, different types can be used if type casting can always be performed safely. There are other forms of GP that address the issue of type safety in a different manner.

Strongly Typed GP (STGP) (Montana, 1995) addresses this issue directly by enforcing that each function and each terminal must declare a type. STGP introduces initialization methods and variation operators that take type safety into account. This guarantees that all individuals generated are type safe. Another way of dealing with the issue of type safety is to represent the possible type combinations in a grammar (e.g., Ryan et al. (1998); Whigham et al. (1995). Besides type safety, evaluation safety must also be assured. Even if types are coherent, some functions might not be applicable for arguments with particular values. To ensure evaluation safety, protected versions of these functions must be provided. The most common case is the need to protect the division operator against division by zero. If the denominator expression evaluates sufficiently close to zero, a protected value is returned instead. This protected value is commonly 1.

The evaluation of a GP individual is similar to the evaluation of other supervised learning models. The individual is provided a set of data instances, and computes the output for each instance. In order to obtain the fitness of the individual, these outputs are then compared with the targets of the learning task.

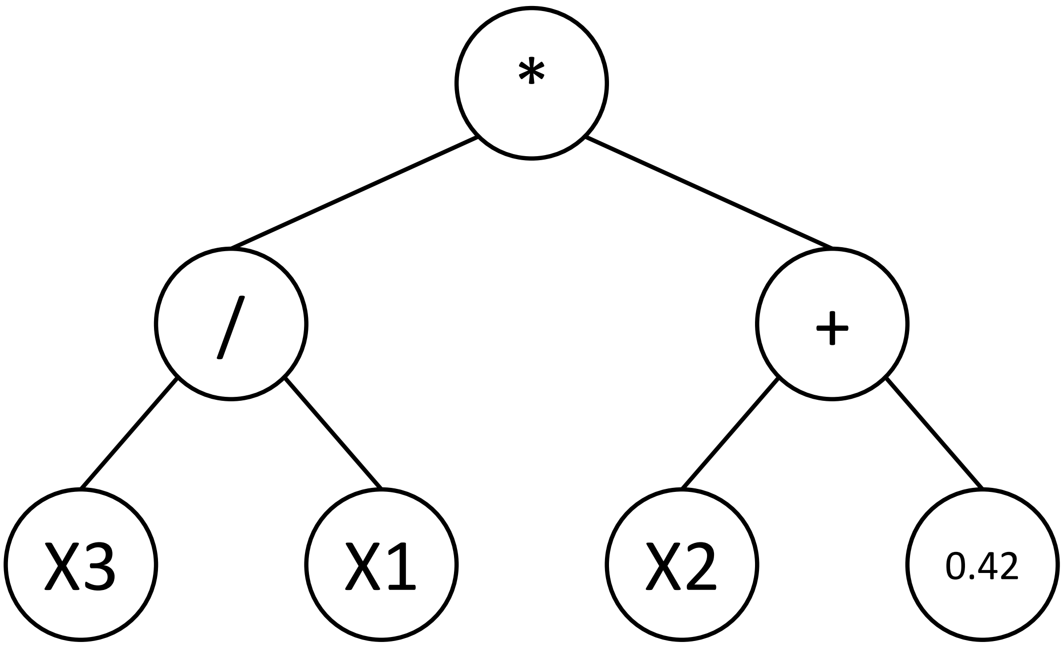


Figure 2.2: A GP individual that represents the function $X3/X1 * (X2 + 0.42)$

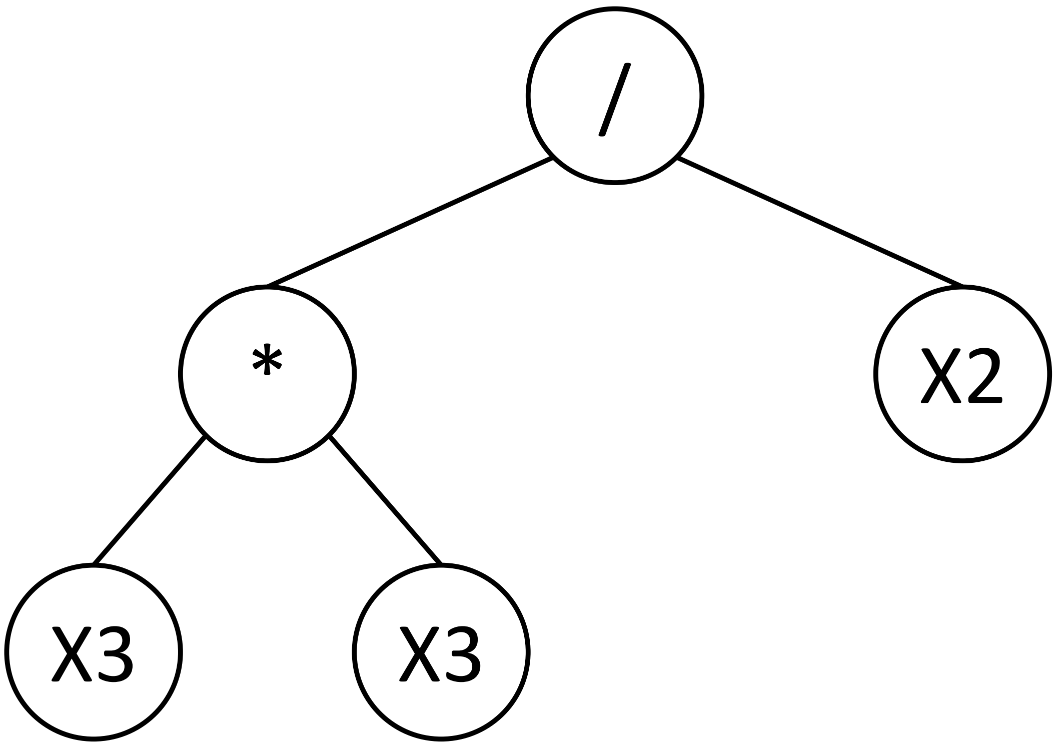


Figure 2.3: A GP individual that represents the function $X3^2/X2$

2.2.2 Initialization

There are three main initialization methods in GP: full, grow, and ramped half-and-half. These methods have associated a maximum tree depth parameter. As the name entails, this parameter limits the depth of each resulting individual. The full method works by adding functions until the maximum tree depth is reached. Each function is selected from the function set with uniform probability. When the maximum tree depth is reached, a terminal must be added to the tree in order not to violate the limit imposed. The terminals are selected from the terminal set with uniform probability. In the full method all terminals will be at the same depth, the maximum depth. This method implies that the trees are full up to a given maximum tree depth, and hence the name of the method. The individual in figure 2.2 could have resulted from the full method with a maximum tree depth of 2. By convention, the depth of the root node of the tree is considered to be 0. The individual in figure 2.3 could not have been initialized with the full method, as there are terminal nodes at different depths. The grow method works by, at each depth, randomly selecting if the node to be added is a function or a terminal. As in the full method, when the maximum tree depth is reached, a terminal must be added to the tree in order not to violate the limit imposed. The trees resulting from the grow method have a higher shape diversity than the ones created with the full method. Both individuals at figure 2.2 and 2.3 could have resulted from the grow method with a maximum tree depth of at least 2. The ramped half-and-half method is a combination of the full and grow methods. The goal is to obtain a more structurally diverse initial population. Ramped half-and-half initializes half of the population with the full method, and the other half with the grow method. The maximum tree depth is varied in order to further structurally diversify the resulting trees.

2.2.3 Selection Operators

Selection operators consist of parent and survivor selection. Parent selection is used to select the individuals in which the variation operators are going to be applied. The selected individuals are known as parents. The most common parent selection method is the tournament selection. Tournament selection works by selecting N random individuals with uniform probability and with replacement. From these initially selected individuals, the best is chosen according to fitness. This best individual is the winner of the tournament. Survivor selection is used to select the new population to be used in the next generation. The most common survivor selection method is the elitist survival. An elitist survivor selection keeps a set of individuals according to fitness, and if more individuals are needed to fill the new population, it selects from the offspring. Perhaps the simplest case of an elitist survivor selection is to the keep the best individual overall, and select offspring to fill the new population. This guarantees that the best individual always survives. An important concept related to selection operators is selective pressure. In this context, selective pressure can be loosely defined as the probability that the best individuals will keep reproducing and/or surviving through the generations. Higher selective pressure results in a higher exploitation of the search space as the search focuses on the current best individuals. This implies a bigger risk of converging to local optima. Lower selective pressure results in a higher exploration of the search space as the search typically focuses on the newly generated individuals. The risk here is that the search might be ineffective, by not being influenced enough by fitness considerations. Finding an appropriate balance in terms of exploration and exploitation is a crucial aspect of EC methods such as GP.

2.2.4 Variation Operators

As other EC techniques, GP uses two variation operators to advance the search: crossover and mutation. The most common GP variation operators are known as standard crossover and standard mutation. Standard crossover can create two offspring from two parents. It firstly randomly selects a point in each parent. These are known as crossover points. These points are root nodes of their own trees, or subtrees. The first offspring is created by copying the first parent but substituting the tree starting at the crossover point, by the tree at the crossover point of the other parent. In other words, the two trees that start at each crossover point are swapped. A second offspring can be created by a similar procedure. This offspring is created by copying the second parent but substituting the tree starting at the crossover point, by the tree at the crossover point of the other parent. Figure 2.4 presents an example of crossover being applied. Standard mutation requires one parent and produces one offspring. Firstly, a random tree is created using the grow or the full methods. Then, a mutation point is selected randomly in the parent. Similarly to the crossover, the offspring is created by copying the parent but substituting the tree starting at the mutation point, by the random tree created. Figure 2.5 presents an example of mutation being applied. Each operator is applied with a given probability. The crossover operator is commonly applied with a much higher probability. Given that these operators may create offspring larger than the parents, a overall growth limit must be imposed. This growth limit is usually applied at the depth of the trees. The most commonly used overall maximum tree depth is 17.

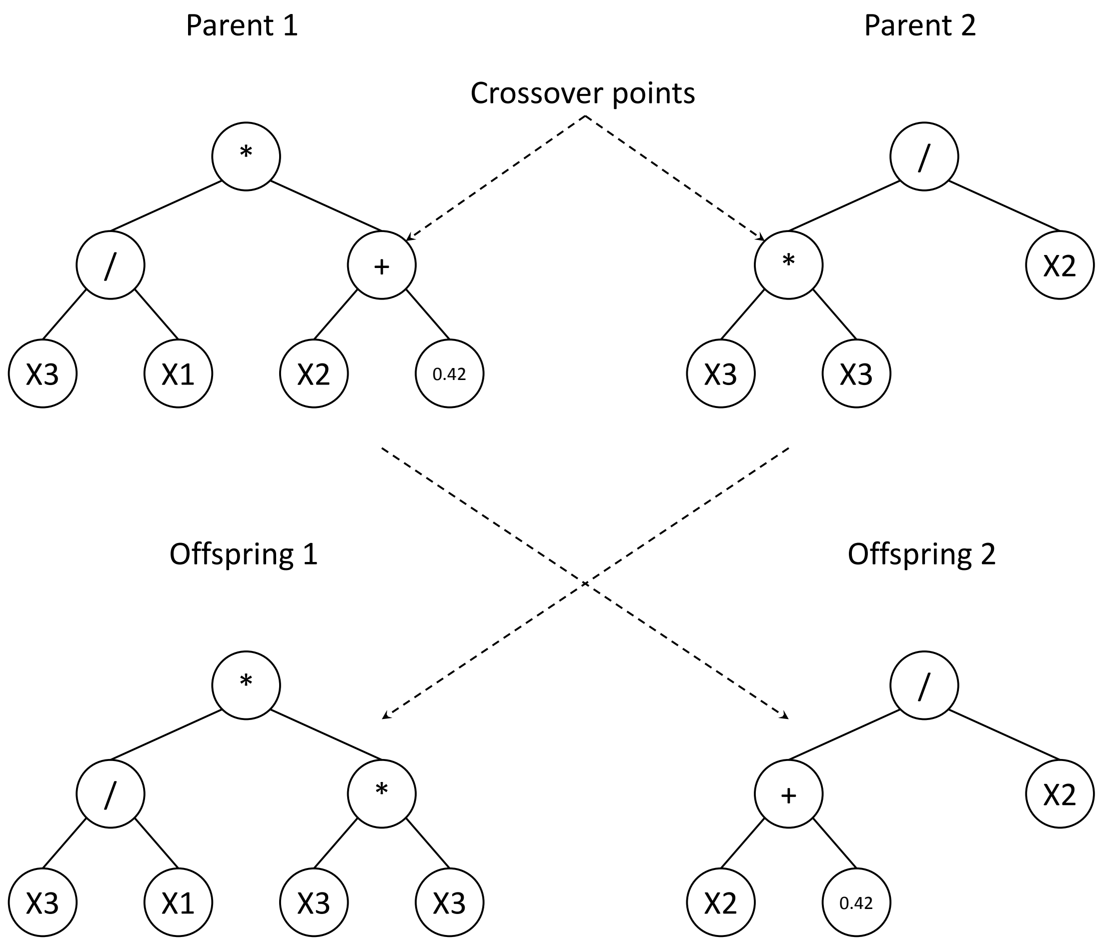


Figure 2.4: An example of a crossover operation

2.3 Generalization in Genetic Programming

The approaches considered in this section are aimed at improving generalization and are focused on Standard GP, with a particular emphasis on regression in real-world datasets.

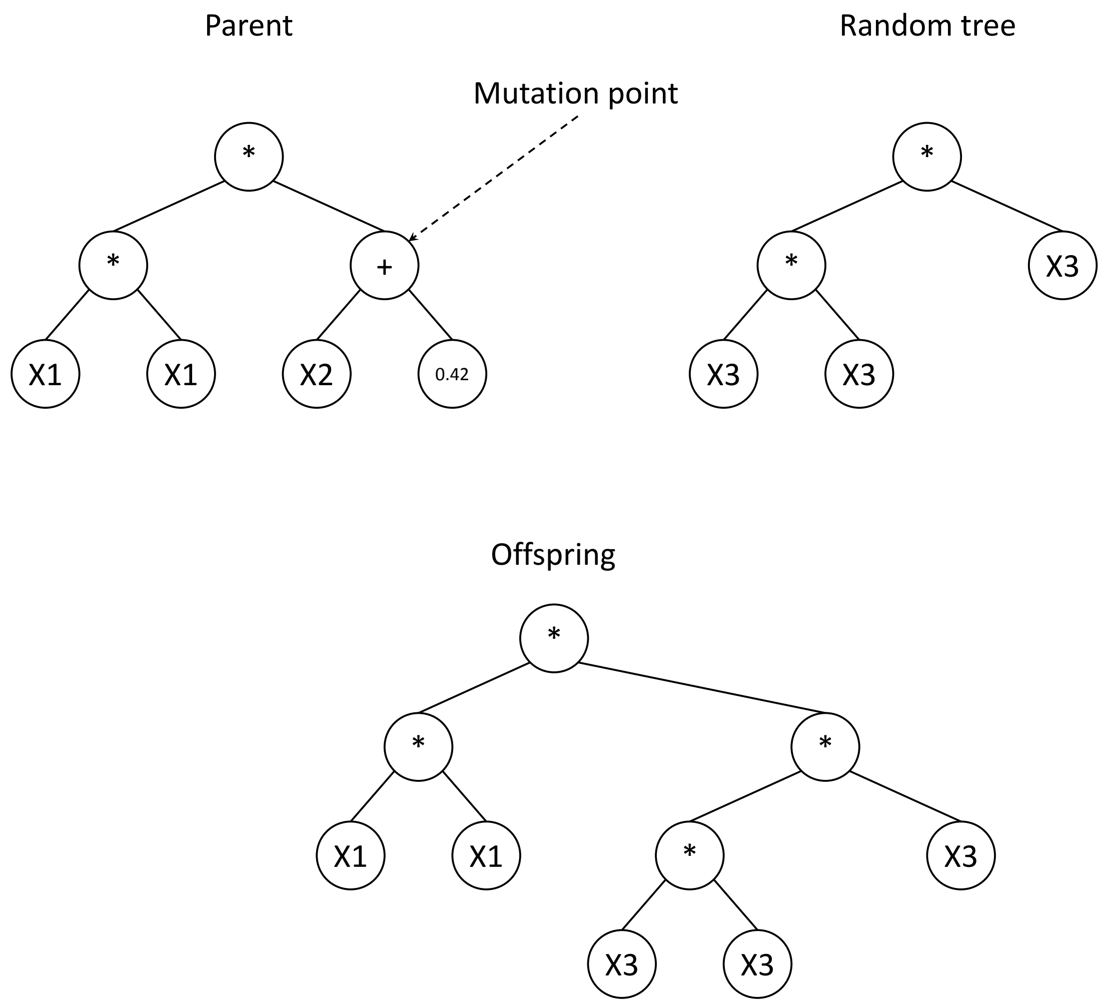


Figure 2.5: An example of a mutation operation

These approaches are grouped in different subsections according to their core notions. The approaches in subsection 2.3.1 share the fact that they bias the search process toward structurally simpler individuals. In these approaches, complexity is usually defined as the size of the trees. Subsection 2.3.2 presents approaches based on functional complexity measures. These complexity measures involve analyzing the behavior of the individuals over a set of data instances, as opposed to focusing on the structure of the individuals. Subsection 2.3.3 describes approaches based on similarities between solutions. Statistical approaches are presented in subsection 2.3.4. Subsection 2.3.5 encompasses a more diverse set of approaches that are not clearly related among themselves or with the other approaches in the remaining subsections.

2.3.1 Structural Complexity

Zhang and Mühlenbein (1995) addressed the relationship between structural complexity and generalization. They did so in the context of using GP to evolve neural networks. Their approach allowed the evolution of the architecture (number of units and layers) and the weights of the neural networks. The fitness function used is a weighted sum of two components: fitting error and complexity. The complexity measure is based on the number of weights, units, and layers of a given neural network. An adaptive balancing of accuracy and parsimony is proposed so that a flexible control of the complexity can be achieved. This allows the search of parsimonious solutions while satisfying the desirable training accuracy. This is accomplished by adaptively changing the complexity weight with respect to the error. The proposed balance encourages fast error reduction at the early stages of evolution while also encouraging stronger complexity reduction at the final stages to obtain parsimonious solutions. Two datasets were used in the experiments: one is artificially generated with noise from the parity function, and the other is a real-world dataset consisting of clinical measurements of 345 different persons. The goal on the latter problem is to classify if a liver is in disorder or not based on some blood pressure measurements. This dataset has 6 input variables. Results show that with the complexity penalty the accuracy is improved both in training and unseen data. Also with the complexity penalty, convergence to the best solution found is three times faster than without the complexity penalty. As expected, solutions found without the complexity penalty are much bigger than those found with the complexity penalty.

Another complexity penalty experiment was conducted by Cavaretta and Chellapilla (1999). This was performed in the context of evolutionary programming using the Australian credit application database. Two algorithms were tested: a no-complexity-bias algorithm (NBC), and a low-complexity-bias algorithm (LCB). The LCB differs from the NBC as it uses a modification in the fitness function meant to penalize larger individuals. As expected, NCB generated considerably larger models than the LCB. However, the generalization error was similar on both approaches. The authors concluded that there is little or no relationship between small model size and lower generalization error.

Becker and Seshadri (2003) proposed adding a complexity penalty factor to the fitness function. The experiments were performed in the context of evolving technical trading rules using GP. They used financial S&P 500 data from January 1954 through December 2002. Data from 1960-1990 were used for training, and data from 1991-2002 were used as unseen data. Results showed that without the complexity penalty factor the results were worse on unseen data than with the factor. The authors concluded that, at least for this domain, reducing complexity does improve the results on unseen data.

Mahler et al. (2005) explored to what extent Tarpeian Control (TC) (Poli, 2003) affects GP generalization. Since TC is able to reduce bloat and hence maintain smaller programs, they argued (in light of the Occam's razor interpretations) that this could help improve generalization. TC is based on the schema theory and aims to slow down the growth of the GP solutions. It achieves this by probabilistically zeroing the fitness of solutions with above average size. This implies that if a solution is above average in size, its fitness can be zeroed if some randomly generated number is smaller than the so called target ratio. This target ratio is a parameter that gives the percentage of over average sized programs that are targeted at every generation. As no general good value is known for the target ratio, it has to be tuned accordingly. Experiments are performed with Standard GP as baseline, and using TC with target ratio values ranging from 10% to 50% with 10% steps. Three regression experiments are conducted. Results show that TC is indeed effective in fighting bloat since it consistently achieves lower average program size without worsening fitness. When it comes to generalization, the results are not consistent. In the first experiment, Standard GP is statistically superior to TC with target ratios of 40% and 50%. However, in the second experiment the reverse happens. In every other case (different target ratios and/or experiment 3) there are no statistically significant differences.

Gagné et al. (2006) tested two approaches to improve GP generalization. These are: the usage of a validation set (three datasets methodology), and the application of parsimony pressure in order to reduce solution complexity (size). Parsimony pressure is achieved by using the lexicographic parsimony pressure method (Luke and Panait, 2002). Experimental testing is conducted in 6 binary classification datasets taken from the Machine Learning Repository at UCI (Lichman, 2013). 10-fold cross-validation is used and care is taken to balance the number of instances of each class between the folds. Four approaches were tested: baseline, with validation (V), with parsimony pressure (P), and with validation and parsimony pressure (B). Results show that there is no real improvement in the generalization over the baseline approach. Results are very similar for all approaches. The real difference happens in the mean size of the solutions. B, V, and P achieve always smaller size solutions than the baseline approach. Furthermore, the B approach achieves over 50% reduction in size when compared to the baseline approach in all but one dataset. V achieves this in 3 out of 6 datasets, and P in one dataset. They found that considerably smaller solutions do not achieve better generalization than the other much bigger solutions. The generalization between the two types of solutions was similar.

2.3.2 Functional Complexity

Vladislavleva et al. (2009) addressed the issue of measuring and controlling the complexity of the models as a way to achieve higher generalization. The proposed complexity measure is called order of non-linearity and adopts the notion of the minimal degree of the best-fit polynomial, approximating an analytical function with a certain precision. A new scheme based on the alternation of several optimization objectives is also proposed. A two objective optimization is performed using the model error and the expressional complexity alternated at each generation with the model error and the order of non-linearity. The authors define two different directions in determining the qualitative complexity of a GP model: complexity of the model expression (compactness of the genotype), and behavior of the associated response surface (smoothness of the phenotype). The expressional complexity is related with the tree size, and it is defined as the sum of the number of nodes in all subtrees of a given tree. This favors trees with fewer layers and, hence, with fewer nested functions as opposed to deep unbalanced trees. The main objective behind the proposed complexity measure is to favor smooth and extrapolative behavior of the response surface, and to discourage highly non-linear behavior, which is unstable toward minor changes in inputs and is dangerous for extrapolation. This complexity measure is based on the degree of the approximating polynomial, and is used as an order of non-linearity of the response surface of the original function (solution). This order can be seen as a numeric value of the deviation of the response surface from a linear hyperplane. A Chebyshev polynomial approximation is used as it has a good treatment of steep response surfaces, and it provides a stable and reliable framework. Experimental testing was conducted in artificial regression datasets as a way to generate noise-free testing data for interpolation and extrapolation. Eight functions were chosen. A ninth problem was introduced based on a real-world dataset (modeling gas chromatography measurements). Pareto GP was the chosen framework. A total of 3 Pareto GP versions were tested. All of them with two optimization criteria, with the last version alternating the second criterion at each generation. The versions tested are the following:

Version 1: Pareto optimization of the sum of squared errors and the expressional complexity.

Version 2: Pareto optimization of the sum of squared errors and the order of non-linearity.

Version 3: Pareto optimization of the sum of squared errors and the expressional complexity, alternated with the Pareto optimization of the sum of squared errors and the order of non-linearity at each generation.

The authors choose version 3 as opposed to using a three objective optimization on the grounds that such a configuration would be very sensitive to the particular linear coefficients and that it would scale badly. They argued that their approach overcomes the curse of dimensionality of the objective space by alternating the two objective optimizations. No comparison against Standard GP was performed. The following results are based on all final solutions over all the 2500 runs. Version 2 outperforms version 1 in terms of testing error on all problems, with statistical significance. These results come at the expense of higher expressional complexity. Solutions generated by version 2 mostly consist of simple operators but are bulky and difficult to interpret. This expressional complexity is largely reduced in version 3. Version 2 significantly outperformed version 3 in terms of testing error in 8 out of 9 problems. The testing error of version 3 is significantly smaller in comparison with version 1 on all problems. The authors note that version 2 and 3 achieve significantly smoother solutions than version 1. They conclude that the proposed complexity measure is effective in significantly increasing generalization.

Vanneschi et al. (2010) proposed a functional complexity measure based on the concept of curvature. The curvature of a function can informally be defined as the amount by which its geometric representation deviates from being straight. The exact calculus of the curvature may be impossible, and an approximation would require the approximating polynomial of that function (as seen in the previous approach). To avoid this, the authors have expressed the complexity of a function by counting the number of different slopes. Higher weights are assigned to inversions in the slope sign. In this work the proposed complexity measure was not used to somehow bias the search. It was only used to discover the relationship between bloat and overfitting by reporting the complexity for each best individual along the evolution. After conducting the experiments, the authors found an unclear relation between the complexity measure, the training fitness, and overfitting.

2.3.3 Similarity Based Approaches

Murphy and Ryan (2008b) explored the generalization ability of Hereditary Repulsion (HR) (Murphy and Ryan, 2008a), and the effect of a particular constraint used within HR. HR works by using a tournament to select the individual with the smallest hereditary overlap in relation to a randomly selected individual. Crossover is then applied to these two selected individuals. This mechanism is intended to avoid premature population convergence. A constraint used within HR enforces that only offspring that are better than both parents can enter the population. This constraint was also tested in a Standard GP approach that does not use HR. The generalization ability of HR and its associated constraint was assessed in 5 different learning tasks: 3 real-world binary classifications from the Machine Learning Repository at UCI (Lichman, 2013), and 2 artificial regressions. A general improvement in terms of generalization was achieved in both approaches.

Vanneschi and Gustafson (2009) proposed a crossover based similarity measure aimed at improving GP generalization by avoiding solutions similar to already known overfitted solutions. The proposed method (repGP) keeps a list of overfitted individuals (called repulsors) and prevents any new individual to enter the next generation if they are similar to any of the known repulsors. This similarity can be determined by the usage of one of the two following operator-based distances: structural (or edit) distance (ED-repGP), and subtree crossover similarity distance (SCD-repGP). Edit distance (ED) works on structural or syntactical similarities, while subtree crossover similarity distance (SCD) works on the probability that some solution can generate a repulsor by means of one crossover operation with another solution from the population. RepGP identifies a solution as a repulsor if that solution is the best on the training set but its performance on the validation set is inferior to the average performance of the best *E* (problem parameter) solutions on the same validation set. Given the previous definition, at the most only one repulsor can be added per generation (since the rule can only be applied to the best solution on the training set). When the repulsors list is empty, repGP is equivalent to Standard GP. Otherwise, tournament selection is applied using the average dissimilarity to the existent repulsors as a criterion to be maximized, i.e., solutions with less similarities to the repulsors are more likely to be selected for reproduction. Experiments are conducted in a real-world dataset (human oral bioavailability) using 2 benchmarks. These benchmarks differ only in the partitioning of the training and testing sets. This partitioning is done randomly in both cases. Standard GP is the chosen baseline technique. Results show that repGP improves upon the Standard GP generalization on the two benchmarks. Also, SCD-repGP achieves better results than ED-repGP. RepGP also achieves smaller solutions than Standard GP. It is interesting to note that the average size of the repulsors is much larger not only than repGP but also than Standard GP. The authors argue that this hints that repGP is really using as repulsors solutions that overfit, arguing that this relates to the MDL principle.

The Semantic Similarity based Crossover (SSC) was proposed by Uy et al. (2010). SSC is based on the Sampling Semantics Distance (SSD) between two trees (or subtrees), which is calculated by choosing *N* random points (data instances), and calculating the mean absolute difference between each corresponding points on the two trees. The authors argue that the exchange of subtrees is most likely to be beneficial if the two subtrees are not too similar or too dissimilar. To ensure this, a lower and an upper bound define the range where the semantic similarity between two trees is valid. These bounds are predefined constants and their best values might be problem dependent. The crossover operator works by choosing two random crossover points, and by checking the semantic similarity between the two. If the semantic similarity is valid, the crossover is applied normally. Otherwise, two new crossover points are selected and the process is repeated. There is a threshold for the total number of tries to apply the operator. If this threshold is reached, two random crossover points are selected and the crossover is applied disregarding the semantic similarity. Four configurations were tested: standard crossover (SC), semantic similarity based crossover (SSC), and SC and SSC with validation (SCV and SSCV respectively). The approaches that use validation perform a division of the whole training set in two: training and validation (67%-33% division). They execute a two objective trial (fitness and size of the individual) in order to extract a set of non-dominated individuals (the Pareto front). These individuals are then evaluated on the validation set, with the best of run individual selected as the one with the smallest error rate on the validation set. Results show that SSC achieves better performance on tests sets than SC. The SCV and SSCV results are slightly worse than SC and SSC respectively. This highlights the fact that the generalization of both methods is not enhanced when a validation set is used. SSC also achieves smaller solutions than SC, which could mean that the control on the semantic level can have positive consequences for the syntactic aspects of the evolved programs. Validation is also shown to help reduce the size of the best individuals.

2.3.4 Statistical Approaches

Nikolaev et al. (2002) proposed several techniques aimed at increasing GP generalization. These techniques are applicable for polynomial representations. The goal of these techniques is to balance the statistical bias and variance. The proposed techniques are: block reformulation, complexity tuning, and a new fitness function. STROGANOFF (Iba et al., 1994) is the baseline GP system in which the new techniques are built on. This system imposes a strong representation bias on the overall polynomials produced at the root. This leads to the overall polynomial order rapidly increasing even if a small number of terms enter the model. To overcome this, a polynomial block reformulation is performed to enable the accommodation of more terms in the overall model without increasing its order. Complexity tuning is achieved by using one of the two proposed techniques: local ridge regression (LRR), and regularized weight subset selection (RWSS). Although computationally stable, LRR is not able to turn weights to zero. RWSS does allow a complete elimination of weights. LRR works on adjusting the polynomial smoothness by manipulating the terms separately since they contribute different curvatures to the model. RWSS computes the significance of each term to the data fitting. All weights that are found to cause improvement of fit above a given threshold remain, while the others are set to zero. The authors define well performing polynomials as those that are: accurate (on training data), predictive, and parsimonious (small). The proposed statistical fitness function has three main components: an accuracy measure that favors highly fit models, a regularization factor that tolerates smoother mappings with higher generalization potential, and a complexity penalty that favors short size models. Experimental testing is conducted on 3 real-world time series. Comparisons are made against Standard GP, the STROGANOFF baseline technique, and a linear model. In all experiments the proposed approach achieved the best generalization results. The improvement in the second time series is particularly dramatic.

Chan et al. (2011) proposed a statistical method called Backward Elimination (BE). BE works by eliminating insignificant terms in polynomials models such as those produced by GP. Each polynomial term is tested for its statistical significance. The insignificant terms are removed until there are none in the corresponding model. It is hypothesized that the elimination of these insignificant terms can lead to solutions with better generalization. The proposed method is called BE-GP, and it is compared with 3 other methods: MAE-GP, ASR-GP, and RAE-GP. These other methods are Standard GP variants with different fitness functions. MAE-GP uses the mean absolute error (MAE) as the fitness function. ASR-GP uses a MDL-based fitness function which is divided into two main components: an accuracy component (proportional to the empirical error over the datasets), and a structural complexity component (proportional to the number of polynomial terms). This fitness function is claimed to be suitable for polynomial modeling as it prevents rapid tree growth, which causes premature convergence to inferior solutions. RAE-GP uses the fitness function from Nikolaev and Iba (2001). This fitness function is similar to the previous fitness function, but it uses the regularized average error (RAE) instead of the average squared residual (ASR). The RAE encourages polynomial models with low-magnitude high-order terms, and with smoother polynomials. These 4 methods are evaluated in 3 real-world manufacturing processes. BE-GP was able to obtain better generalization results than the other methods in almost all tests. The exceptions were: on the first manufacturing process RAE-GP was better in 2 out of 12 tests on the second quality requirement; on the second manufacturing process RAE-GP was better on 4 out of 12 tests; on the third manufacturing process MAE-GP was better in 1 test.

Pennachin et al. (2011) proposed a method based on a technique called affine arithmetic. This technique is a more refined interval method that produces tighter bounds for expressions. Interval methods are techniques for numerical computation in which approximate results are produced with guaranteed ranges or error bounds. They can analyze expressions and produce output bounds as well as detect asymptotes. The proposed method uses the bounds produced by affine arithmetic to detect and discard solutions whose outputs lie outside a desirable range. Also, if some solution contains a possible asymptote then the bounds calculated are infinite, and the solution is eliminated from the population. Experiments are conducted on 15 artificial regression problems, and on 2 real-world time series. Results show that the proposed approach outperforms Standard GP in terms of generalization. A further real-world time series was experimented but in this case the comparison also included Neural Networks and Support Vector Machines. In this last experiment the proposed approach also achieved higher generalization than the other methods. Additionally, Standard GP achieved similar results to Support Vector Machines. Neural Networks presented the worst results. The authors conclude that the proposed approach greatly improves generalization.

2.3.5 Other Diverse Approaches

Robilliard and Fonlupt (2001) applied a method call Backwarding to reduce overfitting when dealing with a real-world dataset: the Photosynthesis Available Radiation (PAR) problem. The method consists in using a validation set and derives its name from the fact that the solutions accepted are not normally those from the final generations of the GP run, since this method goes back as much as needed in the evolution process until the point where overfitting is not yet very relevant. This is achieved by saving two copies of the solutions: one copy for the best solution on the training set, and another copy for the best solution on the validation set. Any new best solution on the validation set is also necessarily the best solution so far on the training set (granted by the algorithm). The opposite does not hold. At the end of the GP process the best saved solution for the validation set is returned. A preliminary evaluation of the Backwarding method is performed, using inverse regression problems with 3 different functions. The results are compared against Standard GP. Backwarding achieves better results on testing and validation sets on all three functions, while generating smaller solutions. This last fact could be interpreted favorably in light of the MDL principle. PAR refers to the number of photons available for photosynthesis in the visible wavelength interval. The interest here is estimating the energy available for photosynthesis from the water-leaving radiance. The inputs for this problem are the selected wavelengths and the expected output is the attenuation coefficient. A new criterion derived from real data is also proposed. Several images were selected and from these 20000 pixels were randomly chosen. These will be the validation set. The desired output for each pixel is not known but it can be bounded (domain knowledge). With this approach the error on the training set is bigger but the models can now generalize better than before.

Paris et al. (2003) compared the generalization performance of 3 GP variants: Standard GP, GP using size fair crossover (Langdon, 2000), and GP with boosting. The experiments are conducted on 2 benchmarks: a regression and a classification dataset. The authors hypothesize that the occurrence of overfitting mainly occurs when too much computing effort is spent, and thus they vary this effort by increasing either the size of the population or the maximum depth allowed. The base overfitting indicator used is the difference between the last generation generalization error ($\mathit{ER}(n)$), and the smallest generalization error ($\mathit{ER}(\mathit{min})$) found during the run ($\mathit{min}$ is the generation where the lowest generalization error occurs). Besides $\mathit{ER}(n)$ and $\mathit{ER}(\mathit{min})$, two other indicators are used. Indicator 1 is the chance that stopping GP at any time after generation min results in an overfitting amount less than 10% (the great the better). Indicator 2 is the number of runs where the overfitting amount at the last generation exceeds a threshold of 10% (the lesser the better). Size fair crossover is intended to slow down the growth of trees during the GP run. The results for this approach are similar to those of Standard GP where in the regression problem there is a small advantage for the smaller depths, and in the classification problem bigger depths are better. The last tested approach is boosting. Boosting is a scheme that can be wrapped around any supervised learning method. It works in a way that the algorithm provides more attention to hard to learn cases. In the regression problem boosting achieves a smaller generalization error than the other approaches. The results on the classification problem are not so good and exhibit a raising generalization error, although the overall precision is more than twice as good as Standard GP.

In the context of financial applications, Chen and Kuo (2003) proposed a measure of degree of overfitting based on the extracted signal ratio. With this measure it is possible not only to conclude if overfitting has occurred, but it is also possible to detect underfitting (lack of performance on the training set). In the case of overfitting, this results in the information (signal) extracted being greater than the maximum information available from the times series. As for underfitting, this results in information being unexploited. In their experiments they noticed a rise in overfitting when the search intensity was raised. In their case, raising the search intensity means increasing the number of individuals in the population. They further confirmed that the search intensity has a positive effect on the degree of fitting, and will consequently contribute to a risk of overfitting. Also, a validation scheme was tried to help to reduce the problem but it did not prove effective.

Liu and Khoshgoftaar (2004) used a method called Random Sampling Technique (RST) to try to reduce overfitting in the context of software quality classification (SQC). RST works by choosing a different subset of the training set (instead of the whole set) at each generation of the GP run. This implies that only solutions that perform well on various different subsets will remain in the population. This method was used before (Gathercole and Ross, 1994) as a way to improve the speed of a GP run, but here the goal is to try to reduce overfitting. The SQC model is used to identify fault-prone (FP) and not fault-prone (NFP) modules based on software metrics and quality data from similar projects or previous system releases. The data used in the experimental setting was collected from a large legacy telecommunications system. Four software releases were considered. A module was considered FP if it had one or more faults, and NFP otherwise. The presented approach used 2 fitness functions: one for the classification error, and another for the solution size. The second fitness function is based on size and biases toward shorter solutions. Two experiments are performed: one with RST, and the other without it (using the entire training set). Overfitting is considered to occur when the predicted error rate for the testing set is greater than 5% of the corresponding error rate for the training set. Results show that there are no overfitted models in release 3 for both experiments. With RST, release 2 presents no overfitted models, and without RST it presents 12 overfitted models out of 15. In release 4, with RST 4 out of 15 models are overfitted, while without RST 13 out of 15 are overfitted. This leads to the conclusion that when RST is in place, the GP models are less prone to overfitting and generalize better. Further similar tests reinforce this conclusion.

The idea behind Canary Functions, proposed by Foreman and Evett (2005), is to measure overfitting during the run by using a validation set. When overfitting starts to occur the search process is stopped. The hope is that by stopping the search as soon as overfitting starts to occur, an increase of the generalization error can be avoided, and thus solutions with lower generalization errors might be achieved. This idea is based on the argument that the effects of overfitting can be reduced by limiting the amount of time spent on training. The definition of a Canary Function is that this function should be, when compared to the fitness function: distinct but related toward meeting the same goal, and its value should differ from the fitness function as overfitting starts to occur. Canary Functions are used as an abstraction for implementing a cross-validation scheme. This scheme has been successful in overcoming overfitting in neural networks and decision trees. Canary Functions call for a need of online indicators that signal when the performance of the solutions on the validation set starts to diverge. Three online indicators were used. The first two were proposed by Prechelt (1998) and are: the generalization loss, and the productivity quotient. The last indicator is the correlation coefficient, which is used to obtain the correlation between the best training set individual and the best validation set individual. Regression was the problem domain used for the experiments. During the experiments, 4 types of overfitting were identified: dramatic divergence, slight divergence, negative performance, and negative performance with crossover. The authors conclude that Canary Functions can be used to determine effectively when overfitting has started to occur.

Da Costa and Landry (2006) proposed an idea to relax the training set by allowing a wider definition of the desired solution. This translates into considering not only the desired output (*y*) correct, but allowing a more broader range to be considered, i.e., it also considers any output in the range [*ymin*, *ymax*] as correct. It is hypothesized that this relaxation can help reduce the generalization error. The experimental testing is performed on a regression dataset. Nine different values were tested for the relaxation, ranging from 0.125% to 20% of the width of the interval occupied by the polynomial. The fitness of an individual is its distance to the middle point of each of the relaxed intervals, except if a particular output belongs to the relaxed interval, in which case its distance is 0. The results show that the generalization error reaches its lowest value at 1% relaxation. The average generalization errors are better than those without relaxation until the 10% mark. The authors conclude that there is an advantage in using a little relaxation. It should be noted that the solutions produced with up to 1% relaxation are the biggest in size, which in some way contradicts the MDL principle, since the solutions with the lowest generalization errors are the biggest. For relaxations greater than 1% the solutions found were smaller than without relaxation.

Vanneschi et al. (2007) argued that using GP with a multi-optimization approach can enhance the generalization of the resulting solutions. Their approach uses two other criteria besides the traditional sum of errors (SE). These are: the correlation between outputs and targets (to maximize), and the diversity of pair wise distances between outputs and targets (to minimize). These criteria resulted from the fact that Standard GP was unable to generalize in the real-world regression applications considered. This lack of generalization came from the fact that the solutions considered, although very similar to the target in the training set range, were increasingly divergent on the testing set range. Given this fact and by looking at the solutions retrieved, the authors proposed to extend the GP framework with a multi-optimization approach with the three criteria previously described. The SPEA2 (Zitzler et al., 2001) technique was used for finding and approximating the Pareto-optimal set. The experiments were conducted by comparing Standard GP with the proposed approach (MOGP) in 3 different functions. In all these functions MOGP achieved lower generalization error, and less and with lower magnitude generalization error oscillations during the run. The solutions found by MOGP are also smaller than those from Standard GP.

Vanneschi et al. (2009) studied the influence of adding Gaussian noise to the data, and using a dynamic training set handling. The proposed Gaussian noise works by perturbing the original value of a terminal each time it is evaluated. This implies that the same variable can have slightly different values in different fitness evaluations. The dynamic training set handling consists in dividing the training set in 5 subsets and using only 4 of those to calculate the fitness. At each 5 generations the subset that was left unused is replaced by one of the 4 that is being used. This translates into having a dynamic training set which is modified in a cyclic way. The two new proposed techniques are tested both separately and together, which means that a total of 3 new GP variants are tested. These variants are experimented in 2 real-world oncologic classification datasets. Comparisons are made against Standard GP, Support Vector Machines, MultiBoosting, and Random Forests. Results show that all the GP methods (the 3 variants and Standard GP) are able to achieve better generalization than the best non-evolutionary method in both datasets. The authors state that these results show that GP can be a valuable technique in this context. Results also show that the proposed GP variants present no clear difference when compared to Standard GP. The authors state that, at least for this particular context, these variants are not helpful in improving GP generalization.

Castelli et al. (2010) performed a comparison of the generalization of several different GP frameworks. These are: Standard GP, Standard GP with dynamic operator equalization (DynOpEq) (Silva and Dignum, 2009), and a set of multi-optimization GP variants. NSGA-II (Deb et al., 2002) is the chosen multi-objective evolutionary method. The chosen fitness functions are: the root mean square error (RMSE) between the target and the obtained values, solution size (number of nodes), and the variance of the error between the target and the obtained values. From these, three variants are used: RMSE + size (MOsize), RMSE + variance (MOvar), and RMSE + size + variance (MOsize+var). These three variants are compared with Standard GP and DynOpEq. Experimental testing is conducted on a real-world problem (prediction of pharmaceutical drug toxicity) with high dimensionality. On testing data MOsize achieves the best results. MOvar and MOsize+var results are quite similar to MOsize. The authors argue that the similar results obtained on both the training and the testing sets of the MO approaches, is evidence of the generalization ability provided by multi-optimization. The MOsize+var approach does not perform better than any MO approach with only one auxiliary function. The authors find this counterintuitive but argue that this could result from a low number of generations since they believe that the increase of auxiliary function can slow down convergence. The authors also point out that DynOpEq is the approach that produces smallest individuals, although this fact does not guarantee the absence of overfitting. They state that this hints that bloat and overfitting are two separate phenomena. Statistical validation confirms that multi-optimization improves GP generalization.

Finally, in the context of the Compiling Genetic Programming System (Nordin, 1994), Banzhaf et al. (1996) showed the positive influence of the mutation operator in the generalization achieved. No similar studies are known in Standard GP.

2.4 Discussion

The issues of generalization and overfitting did not receive much attention in the early years of GP. This has recently been changing as these issues become more widely studied. Generalization in GP has been recently considered an important open issue (O'Neill et al., 2010).

2.4.1 Considerations on Theory and Experimental Methodology

Poli et al. (2010) emphasized that the study of the theoretical aspects of generalization in GP has not developed nearly enough in the last decade. Teller and Andre (1997) and Giacobini et al. (2002) have studied the number of data instances needed for GP to learn. However, these approaches do not address directly the issue of generalizing to unseen cases. Outside of GP theory, in Computational Learning Theory, Ehrenfeucht et al. (1989) proved a general lower bound on the number of data instances needed for learning, under the Probably Approximately Correct (PAC) Learning model (Valiant, 1984). Although interesting from a theoretical perspective, the bound on the needed training data is not commonly relevant in real-world scenarios. More commonly, the issue is that the available data is scarce, and no more data is easily obtained. Recently, Mambrini and Oliveto (2016) proved negative results on the generalization of simple GP algorithms in the context of learning particular boolean functions. No similar theoretical results are known for the Standard GP version.

In the early years of GP, approaches aimed at improving generalization were very uncommon. Kushchu (2002) mentioned that at that point, the issue of generalization in GP had not received the attention it deserved. In fact, it was even uncommon to measure the performance in a set of unseen data. Notably, in Koza (1992) most of the problems presented did not use separate training and unseen data, so performance was never evaluated on unseen cases (Kushchu, 2002). This also occurred in the early days of classical Machine Learning (Domingos, 2012). However, in the case of Machine Learning, the models used at the time were relatively simple. The underlying reasoning was that, since those models were simple, they were unlikely to overfit. In this case the bigger risk would be underfitting. In the case of GP, the flexibility and complexity of the individuals was embraced from the start. This is still seen as one of the most distinct characteristics of GP. It is clear from the empirical evidence that constructing highly flexible individuals/models can be risky in terms of overfitting. In this scenario, a correct methodological approach becomes even more critical. Eiben and Jelasity (2002) also mentioned that at that point, it was uncommon to report unseen data results in the larger EC area. They also referred as problematic the unsubstantiated choice of problem instances, and the missing experimental details needed to reproduce the experiments. More recently, Costelloe and Ryan (2009) highlighted the need for more accurate generalization performance reports. By now it is standard practice in GP to properly report the generalization ability of the individuals as measured by the performance in separate set of unseen data.

A still common issue in GP is the usage of trivial artificial datasets. White et al. (2013) addressed this issue by proposing a blacklist of benchmark problems that should be avoided given their simplicity. They mention that an experimental evaluation that only includes these blacklisted benchmark problems is considered inadequate. This consideration is supported by a survey conducted in the GP research community. This community survey originated as a follow-up to an initial study (McDermott et al., 2012) of the problems commonly used in GP. White et al. (2013) also proposed substitute benchmark problems for the problems in the blacklist. However, some issues arise regarding the regression datasets proposed. One of the issues is that the majority (5 out of 7) of these datasets are artificial and defined within the GP community. This reduces the scope of comparison with non-GP methods, as these artificial datasets are not commonly used outside of the GP area. Furthermore, as it commonly happens with artificial regression datasets in GP, they are low dimensionality problems. These artificial regression datasets proposed have at most 5 input variables. Using these datasets may result in the GP community focusing too much on potentially easy problems. Although it is not guaranteed that, as the dimensionality increases, so does the difficulty of the dataset. However, considering the increasing availability of real-world datasets, and their use outside of GP, it would be a missed opportunity not to focus on these real-world datasets (the most notable case is the widely used Machine Learning Repository at UCI (Lichman, 2013)). The adoption of widely used dataset repositories increases the comparability of GP with other supervised learning algorithms. The increased comparability would allow the GP community to better understand which datasets are particularly challenging overall, and which datasets are particularly challenging for GP. This increased comparability could also contribute to a greater acceptance of GP in the larger supervised learning area. Another issue with the some of the regression datasets proposed by White et al. (2013), is the lack of definition of how the unseen data is constructed. White et al. (2013) mentioned that for those datasets, a method for estimating the generalization can still be implemented by using a hold-out set or cross-validation. The issue is that this lack of specification will probably result in different ways of measuring the generalization. If this is the case, then the comparability between methods using the same dataset will be diminished. An agreed upon specification for every dataset proposed would reduce the comparability issues.

During the review presented in the previous section, it was noticed that in some works the training and testing sets contained overlapping data instances. In other words, some sets of data used to estimate the generalization ability contained data instances that were also present in the training data. This usually occurred when using artificial regression datasets. In these works, the training and testing sets were created by randomly sampling from a given range of data instances. This would be methodologically correct if care was taken to avoid repeated instances. In some of the works, this is not the case. This overlapping between the training and testing sets, creates an artificial increase in the generalization estimation.

As a final methodological consideration, there is still some lack of reporting regarding the statistical significance of the results. However, this reporting is becoming much more common. Overall, the GP community has been gradually overcoming the methodological flaws that were prevalent in the early years.

2.4.2 Trends in Genetic Programming Approaches

From the approaches reviewed in the previous section, a clear trend emerges. Overwhelmingly, the initial approaches were focused on the structural complexity of the individuals. However, a change of direction has clearly occurred, as the consideration for structural complexity was replaced by different approaches.

Firstly, it is relevant to understand the reasoning behind the focus on the structural complexity. As seen with other supervised learning methods, the Occam's razor has also been an important source of inspiration in GP. Based on the Occam's razor interpretations mentioned in section 2.1, the general inductive bias in GP was clearly based on minimizing the complexity of the individuals. The reasoning was that if simpler individuals were found, then a better generalization would be achieved. Given that the size of an individual is one of the simplest definitions of complexity, the majority of the initial approaches focused on achieving smaller individuals. However, as seen in the previous section, the results of this type of approaches are inconsistent. In some cases, smaller individuals did indeed generalize better, but the reverse was true in other cases. In essence, the empirical evidence did not support a general claim that smaller individuals generalize better. It might even be possible that other explanation could account for the cases where smaller individuals generalized better. Particularly, it could be the case that the region of the search space containing the smaller individuals was simply being better explored, and not necessarily because it was more likely to find individuals that generalized better in that search space region. As the bias toward shorter individuals did not presented robust results, the focus had to be reconsidered.

The initial focus on the size of the individuals was also related to another issue occurring in GP: bloat. Bloat is said to occur when the individuals continue to grow but no corresponding improvement in fitness is achieved (Silva and Costa, 2009). There was also a considerable focus on the issue of bloat, not only because its occurrence hindered the search progress but also because it was hypothesized that the smaller resulting individuals would lead to a better generalization. Consequently, it was thought that these two issues were related, and that counteracting bloat would lead to positive effects in terms of generalization. This, however, has been recently challenged. Contributions show that bloat free GP systems can still overfit, while highly bloated solutions may generalize well (Vanneschi and Silva, 2009). As it stands, it seems that bloat and overfitting are two independent issues. These results further reduce the focus on achieving smaller individuals as a way to improve generalization.

Since the focus on the structural complexity did not provide satisfactory results, a different set of approaches have been explored. A recent type of approach is based on the definition of different measures of complexity that might be better correlated with the generalization of an individual. These complexity measures are based on the behavior of a given individual on a set of data instances. These are known as functional or behavior complexities. Functional complexity measures arise because the apparent structural simplicity or complexity of an individual might be misleading. For instance, an individual that represents the cosine function is structurally very simple but produces a considerably complex behavior. On the other hand, an individual that represents a sum of several simple terms is structurally more complex, but its behavior is typically less complex. The approaches based on functional complexity in GP, are recent but appear to be promising. The similarity based approaches are also a recent and potentially interesting research area. As seen in the previous section, the most recent approaches are very diverse. This diversity also arises because of the several interacting components involving GP. This leads to a potential large number of different ways of approaching the generalization issue. Because of this, the rather diverse set of recent approaches should not be surprising.

Finally, it should be noted the increasing interest on the connection between GP and the rest of the supervised learning area. Previously there were very few GP works that incorporated non-GP concepts. A few of those exceptions were related with the connection with Ensemble Learning and the bias-variance decomposition (Iba, 1999; Keijzer and Babovic, 2000; Paris et al., 2003). Recently, more GP studies have appeared that incorporate non-GP concepts. Connections with Statistical Learning Theory (Amil et al., 2009; Chen et al., 2016; Montana et al., 2009), and PAC learning (Kötzing et al., 2011) have been recently studied. The usage of GP within other areas has also been increasing (Koza, 2010).

3 Experimental Methodology

This chapter describes the experimental methodology. Section 3.1 presents the datasets. Section 3.2 specifies the parameters used. Section 3.3 describes the statistical tests and presents some general considerations.

3.1 Datasets

Three high-dimensional regression real-world datasets are used in the experiments. They are referred to as: Bio, PPB, and LD50.

In the Bio dataset the goal is to predict the human oral bioavailability of a given pharmaceutical drug. The human oral bioavailability is a pharmacokinetics parameter that measures the percentage of the initial orally submitted drug dose that effectively reaches the systemic blood circulation after passing through the liver. Being able to reliably predict the bioavailability value for a potential new drug is outstandingly important, given that the majority of failures in compounds development from the early nineties to nowadays are due to a wrong prediction of this pharmacokinetics parameter during the drug discovery process (Kennedy, 1997; Kola and Landis, 2004). This dataset consists of 359 data instances, where each instance has 241 attributes that describe the molecular structure of a candidate drug.

In the PPB dataset the goal is to predict the value of the plasma protein binding level of a given pharmaceutical drug. The plasma protein binding level quantifies the percentage of the initial drug dose that reaches the blood circulation and binds to the proteins of plasma. This measure is fundamental for good pharmacokinetics, both because blood circulation is the major vehicle of drug distribution into human body and since only free (unbound) drugs can permeate the membranes reaching their targets (Archetti et al., 2007). This dataset consists of 131 data instances, where each instance has 627 attributes that describe the molecular structure of a candidate drug.

In the LD50 dataset the goal is to predict the median lethal dose of a given pharmaceutical drug. The median lethal dose refers to the amount of compound required to kill 50% of the considered test organisms (cavies). Reliably predicting this and other pharmacokinetics parameters would permit to reduce the risk of late stage research failures in drug discovery, and enable to decrease the number of experiments and cavies used in pharmacological research (Archetti et al., 2007). This dataset consists of 234 data instances, where each instance has 626 attributes that describe the molecular structure of a candidate drug.

3.2 Baseline Parameters

The following parameters are the baseline parameters used in the experiments. Unless stated otherwise, all experiments use these same parameters. The parameters are provided in table 3.1. Furthermore, fitness is computed as the Root Mean Squared Error (RMSE) between the outputs of an individual and the targets of the dataset. For each run, a different randomly selected data division is performed. Each method uses the same data division for equivalent runs.

3.3 Statistical Tests and Other Considerations

Claims of statistical significance are based on Mann-Whitney U tests, with Bonferroni correction, and considering a significance level of $\alpha = 0.05$. A non-parametric test is used because the data are not guaranteed to follow a normal distribution. An important tool in the analysis of GP are evolution plots. Evolutions plots allow an analysis of the evolution of a given measure throughout the generations. All evolution plots presented in the next chapters are based on the median values over 30 runs of some particular measure. The median is preferred over the average as it is more robust to outliers. Training error

Table 3.1: Baseline parameters

Parameter	**Value**
Runs	30
Population	500
Generations	200
Data partition	Training 50% - Unseen 50%
Crossover operator	Standard subtree crossover, probability 0.9
Mutation operator	Standard subtree mutation, probability 0.1, new branch maximum depth 6
Tree initialization	Ramped Half-and-Half, maximum depth 6
Function set	+, -, *, and / (protected)
Terminal set	Input variables, constants -1.0, -0.5, 0.0, 0.5 and 1.0
Parent selection	Tournament selection of size 10
Survivor selection	Best individual always survives
Maximum tree depth	17

evolution plots are based on the training error of the best individual at each generation. The error on unseen data is referred to as generalization error. Generalization error evolution plots are based on the generalization error of the best individual selected according to the training error. Tree size and tree depth evolution plots present the tree size or depth of the best individual selected according to the training error. Tree size is computed as the number of nodes of an individual.

4 Training Data Sampling Strategies

This chapter explores how different training data sampling strategies can be used to potentially improve the resulting generalization. Section 4.1 studies how the use of dynamic subsets of training data influences the generalization outcome. Of particular interest is the usage of small subsets of training data. Sections 4.2 and 4.3 explore different ways of balancing the usage of different amounts of training data.

4.1 The Generalization Effect of Using Small Dynamic Subsets of Training Data

4.1.1 Dynamic Subsets of Training Data

In this section, the dynamic use of training data is performed by applying the Random Sampling Technique (RST). This approach is also known as random subset selection. In the RST, a new subset of training data is randomly selected at the beginning of each generation. The size of the subsets is a RST parameter. The fitness of each individual is recomputed for each new training data subset. Parent and survivor selection are performed by taking into account the fitness computed in the training data subset. The whole training data is never used directly. This implies that only individuals that perform well on various different training data subsets will be able to reproduce and/or remain in the population. It is expected that the best surviving individuals have captured the underlying relationships of the data instead of overfitting it.

The RST was previously shown to have positive effects. Gathercole and Ross (1994) showed that this type of approach could improve the speed of a GP run, while still achieving similar results as using all of the training data at each generation. Gathercole and Ross used between 10% and 15% of the total training data depending on the experiment. Liu and Khoshgoftaar (2004) were able to reduce overfitting in a software quality classification task by applying RST with 50% of the total training data.

4.1.2 Results

In this section, RST is used in the high-dimensional real-world datasets previously presented. Different sizes of training data subsets are tested. These sizes are obtained from a given percentage of the total training data. The percentages tested are: 5%, 25%, 50%, 75%, and 95%. Given the particular interest in studying the use of small subsets, the extreme case of using only a single training instance at each generation is also tested. This configuration is simply labeled RST. Notice that, as usual, the Standard GP baseline uses all the training data at each generation. Figure 4.1 presents the training and generalization errors evolution plots for Standard GP and RST. For an easier interpretation of the gap between the training and generalization errors, a simple measure of overfitting is used. This measure is computed as the absolute difference between the training and generalization errors. Figure 4.2 presents the overfitting evolution plots with the described measure.

Starting with the training error, the behavior across all variants is as expected. The variants that use more training data (Standard GP and RST with higher percentages) are able to achieve lower training errors. This is an intuitive result as presenting more data to a search method allows it to learn faster. Notice that as the percentage of training data increases in RST, the closer the training error behavior is to Standard GP. In the RST setting that uses the most training data (RST 95%), the behavior is almost the same as Standard GP. However, the variants that fit the training data faster are incurring in overfitting. The generalization behavior evolutions show that, after a certain point, the best training data performing variants are simply starting to overfit. As is clear in Standard GP and in the RST variants with higher percentages of training data usage (50% or above), the bigger training error reduction comes at the expense of the resulting generalization. By the end of the runs, all of these variants present significant overfitting. On the other hand, the

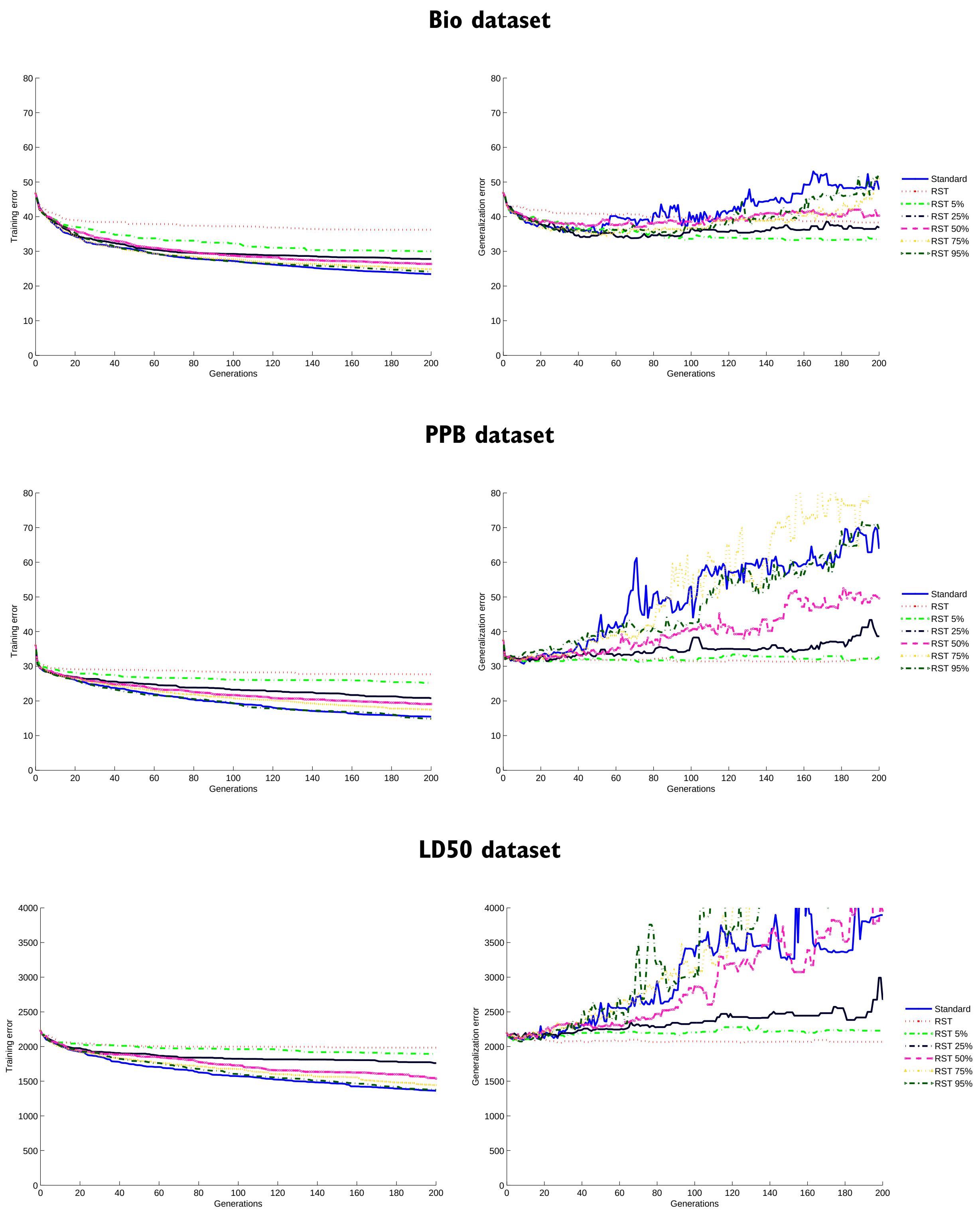


Figure 4.1: Training and generalization errors evolution plots for Standard GP and RST with different parameters

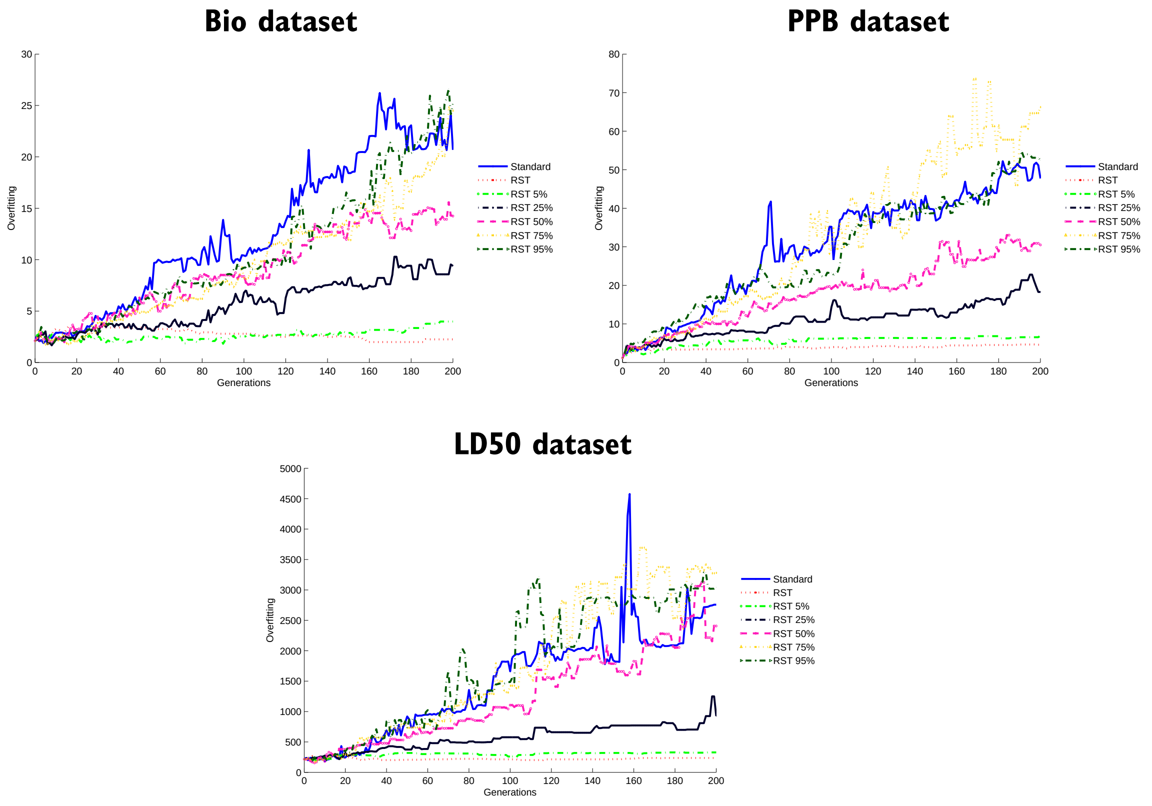


Figure 4.2: Overfitting evolution plots for Standard GP and RST with different parameters

RST variants that use very small percentages of the training data (RST and RST 5%) are able to avoid overfitting. These variants achieve the best median generalization errors across all datasets. RST is the best performing variant in the LD50 dataset, while RST 5% achieves the best median generalization error in the Bio dataset. In the PPB dataset, RST and RST 5% achieve similar generalizations. In RST and RST 5%, the reduction of the training error usually signifies a similar reduction in the generalization error. This behavior is clear in the mentioned overfitting plots (figure 4.2). The overfitting values of RST and RST 5% are relatively stable from the beginning of the runs. In Standard GP and in the RST variants with higher percentages of training data usage, the overfitting values reflect the constantly widening gap between training and generalization errors. Similarly to the training and generalization errors evolution, as the percentage of training data increases in RST, the closer the overfitting evolution is to Standard GP.

Figure 4.3 shows the boxplots with the generalization errors achieved in the last generation of each variant. Besides achieving better median generalization errors, RST and RST 5% also present more robust generalizations across different runs. Standard GP and the RST variants with higher percentages of training data usage present considerable variations on the generalization achieved depending on the specific run. Focusing the analysis on RST and RST 5%, both variants generalize better than Standard GP, with statistical significance, in the PPB (p-values: RST $\mathbf{1.335 \times 10^{-6}}$, and RST 5% $\mathbf{1.150 \times 10^{-6}}$) and the LD50 dataset ($p$-values: RST $\mathbf{1.111 \times 10^{-7}}$, and RST 5% $\mathbf{1.067 \times 10^{-6}}$). In the Bio dataset, only RST 5% is superior with statistical significance (p-value $\mathbf{7.899 \times 10^{-5}}$).

Another advantage of using small dynamic subsets of training data is the smaller resulting individuals. Figure 4.4 presents the tree size evolution plots, and figure 4.5 presents the tree depth evolution plots. The best individuals found in RST and RST 5% are considerably smaller than the ones produced by Standard GP. The tree size evolution plots show that the tendency is to reach bigger individuals as the amount of training data provided increases. In terms of tree depth, RST achieves particularly shallow trees. Almost all of the other variants reach the imposed tree depth limit.

Across the tested datasets, the general trend shows that the use of small dynamic subsets of training data prevents the appearance of overfitting, and contributes to a robust generalization outcome. These outcomes are even possible when using a single training instance at each generation. Besides improving generalization, the resulting individuals are also smaller. A further advantage is that this type of approach does not directly constraint

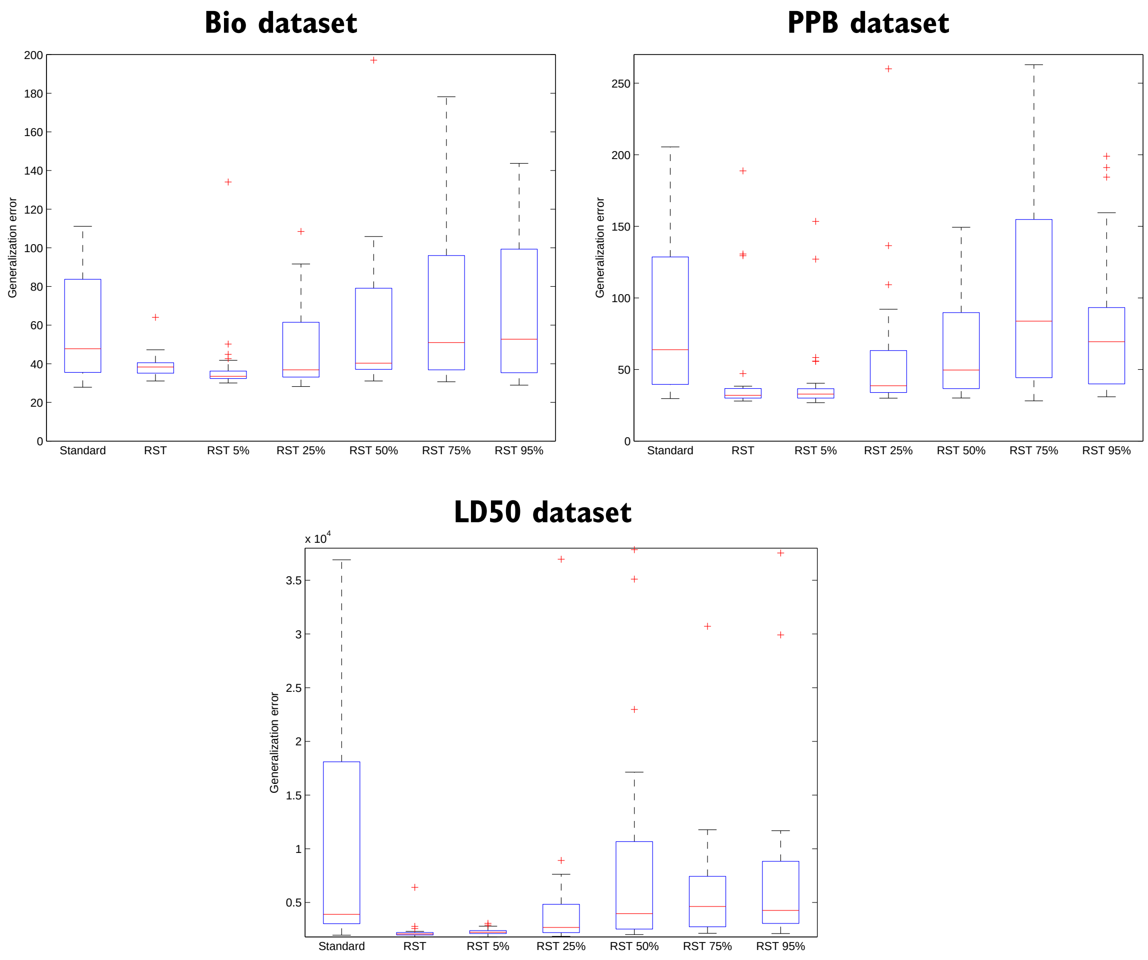


Figure 4.3: Generalization error boxplots for Standard GP and RST with different parameters

the flexibility of the resulting GP individuals.

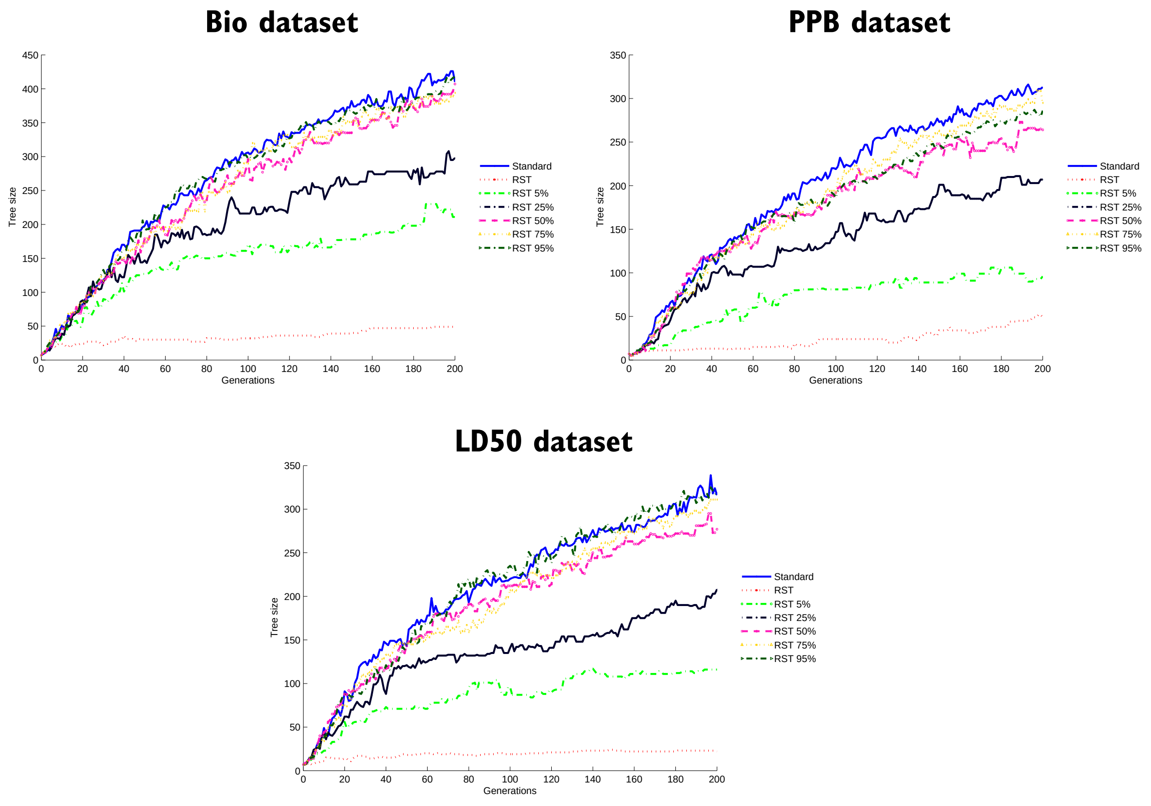


Figure 4.4: Tree size evolution plots for Standard GP and RST with different parameters

4.2 Deterministic Interleaved Sampling

Given the particularly interesting results of using a dynamically changing single training instance, this section and the next explore how the usage of a single training instance can be balanced with periodically using all of the available training data. The rationale behind this approach is based on trying to keep overfitting low (represented by using a single training instance), and still presenting enough information so that a general pattern can be found (represented by using all training data). The usage of all training data might also increase the learning rate of the search method. This type of approach is named interleaved sampling as it interleaves the usage of the training data between a single training instance and all the training instances. In other words, in some generations the quality of

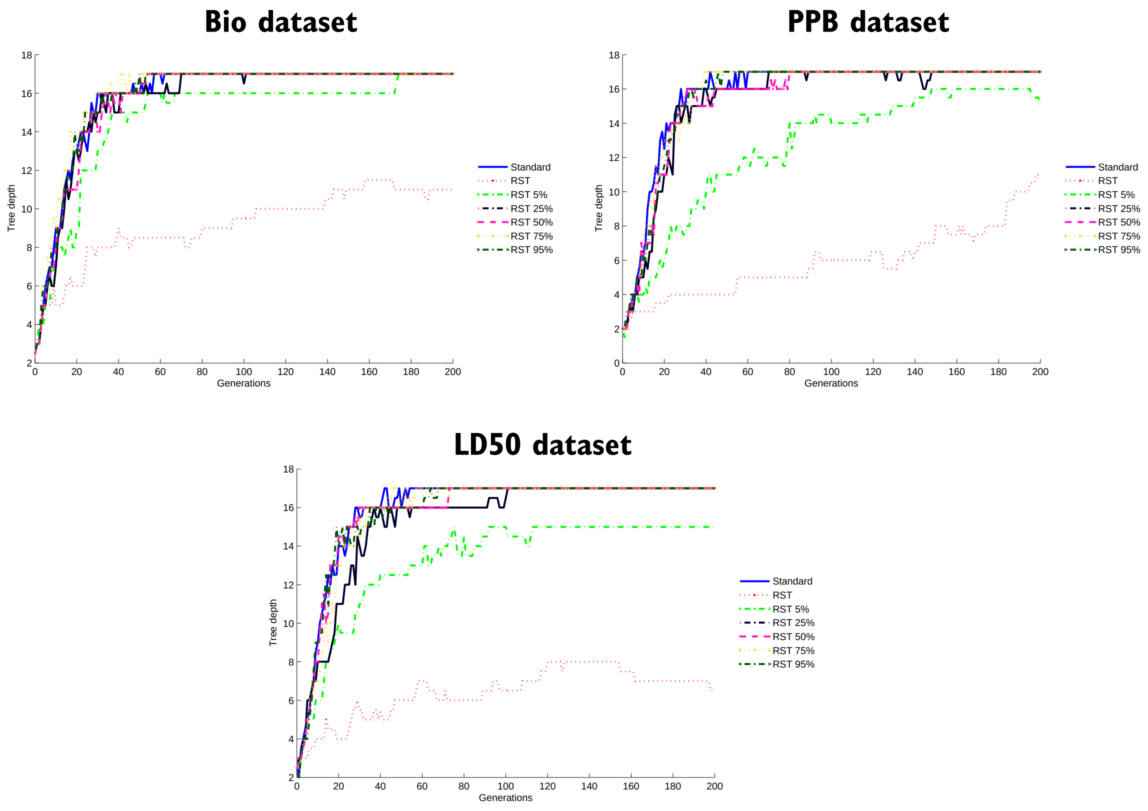


Figure 4.5: Tree depth evolution plots for Standard GP and RST with different parameters

the individuals is computed based on a single randomly selected training instance, while on the remaining generations the quality of the individuals is computed based on all of the available training instances. This section studies two deterministic interleaved sampling approaches. In this context, deterministic means that the specific generations in which a single training instance is used are known a priori. The next section studies a probabilistic approach of choosing in which generations a single training instance is used.

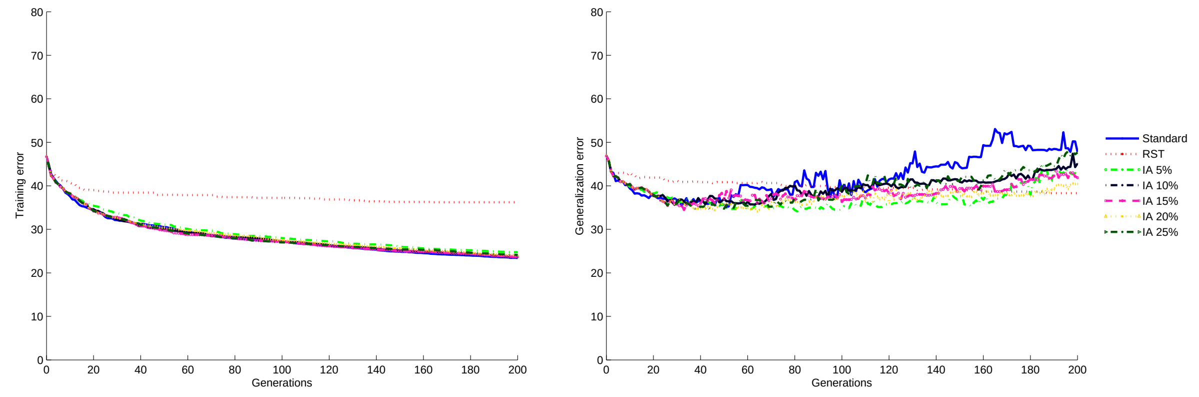
4.2.1 Interleaved All

The first deterministic interleaved sampling approach is named Interleaved All (IA). IA entails a preference to using all the training data available. In other words, in the majority of the generations the quality of the individuals is computed based on all of the available training data. A parameter is added in order to define how many generations using all training instances are performed for each generation where a single training instance is used. This parameter is defined by a given percentage of the total number of generations. For instance, if this parameter is set to 5% of the total number of generations, this means that (with the total number of generations being considered experimentally) the first generation uses a single training instance, while the following 10 generations (5% of 200) use all the training instances. The next generation reverts back to using a single training instance, and the following 10 generations use all the training instances. This process is repeated until the total number of generations is reached. The values tested for this parameter are: 5%, 10%, 15%, 20%, and 25%. Notice that 25% is already a considerably high value as this implies that in total only 4 generations will use a single training instance.

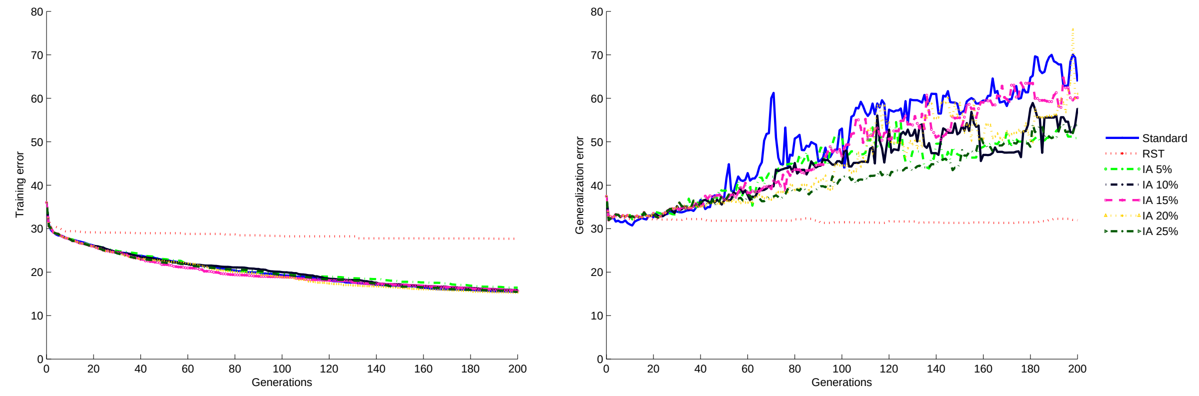
Figure 4.6 presents the training and generalization errors evolution plots for Standard GP, RST, and IA with the different parameters tested. Across all datasets, the behavior of the IA variants is very similar to the behavior of Standard GP, both on training and generalization. As Standard GP, the IA variants are effective in reducing the training error, but do so by incurring in overfitting. All IA variants result in considerably higher median generalization errors in comparison with RST. In terms of tree size (figure 4.7) and tree depth (figure 4.8), the IA variants also behave similarly to Standard GP. The resulting individuals are considerably bigger and deeper than the ones produced by RST.

These results show that an approach based on using a majority of the generations with the quality of the individuals been computed based on all of the available training

Bio dataset



PPB dataset



LD50 dataset

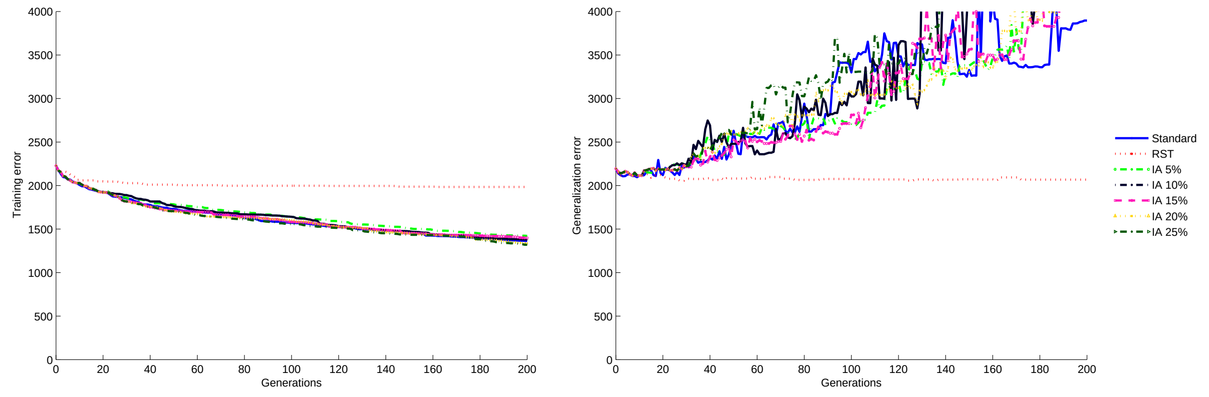


Figure 4.6: Training and generalization errors evolution plots for Standard GP, RST, and IA with different parameters

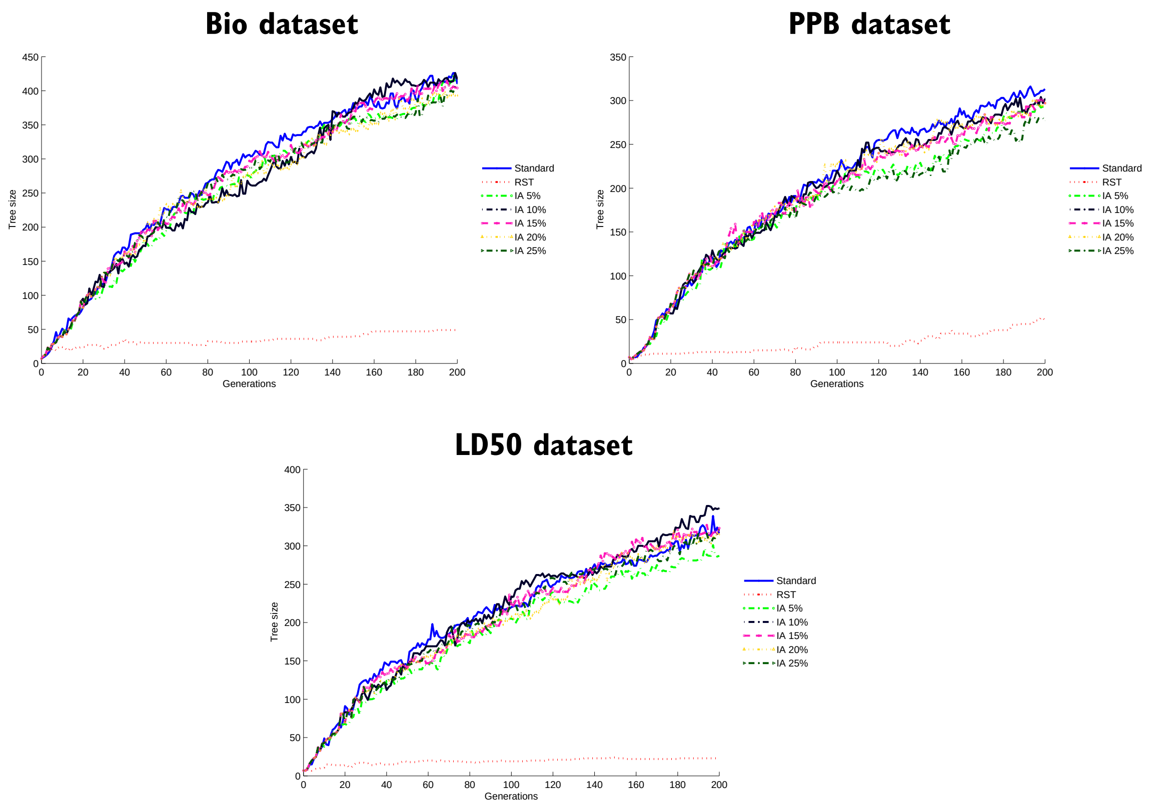


Figure 4.7: Tree size evolution plots for Standard GP, RST, and IA with different parameters

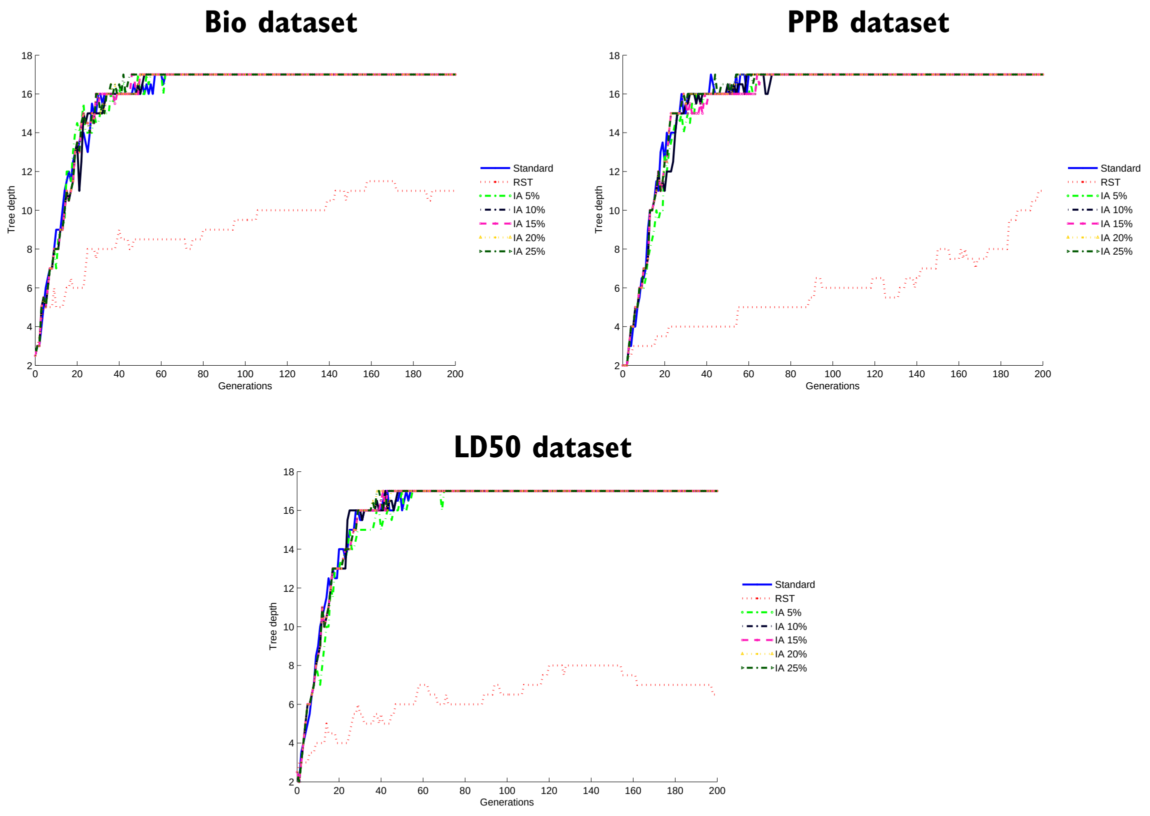


Figure 4.8: Tree depth evolution plots for Standard GP, RST, and IA with different parameters

data is not effective in reducing overfitting, even with a periodical usage of a single training instance. It seems that a more predominant usage of generations with a single training instance must be consider in order to avoid overfitting.

4.2.2 Interleaved Single

The second deterministic interleaved sampling approach is named Interleaved Single (IS). IS entails a preference to using a single training instance. In other words, in the majority of the generations the quality of the individuals is computed based on a single training instance. A parameter similar to the one used in the IA approach is added. In the case of IS, the parameter defines how many generations using a single training instance are performed for each generation where all the training instances are used. For instance, if this parameter is set to 5% of the total number of generations, this means that (with the total number of generations being considered experimentally) the first generation uses all the training data, while the following 10 generations (5% of 200) use a single training instance. The next generation reverts back to using all the training data, and the following 10 generations use a single training instance. This process is repeated until the total number of generations is reached. The values tested for this parameter are the same as in the IA approach.

Figure 4.9 presents the training and generalization errors evolution plots for Standard GP, RST, and IS with the different parameters tested. The behavior of the IS variants is very similar to the behavior of RST, both on training and generalization. Because of this similarity, the IS variants present the same generalization advantage over Standard GP as RST does. The same similarity with RST occurs in terms of tree size (figure 4.10) and tree depth (figure 4.11). As RST, the IS variants achieve much smaller and shallower individuals in comparison with Standard GP. Despite these advantages over Standard GP, the IS variants are not able to surpass the RST outcomes.

These results show that an approach based on using a single training instance to compute the quality of the individuals in a predominant number of the generations is effective in avoiding overfitting. However, the effect of presenting all training data to the search method with the periodicity tested is negligible.

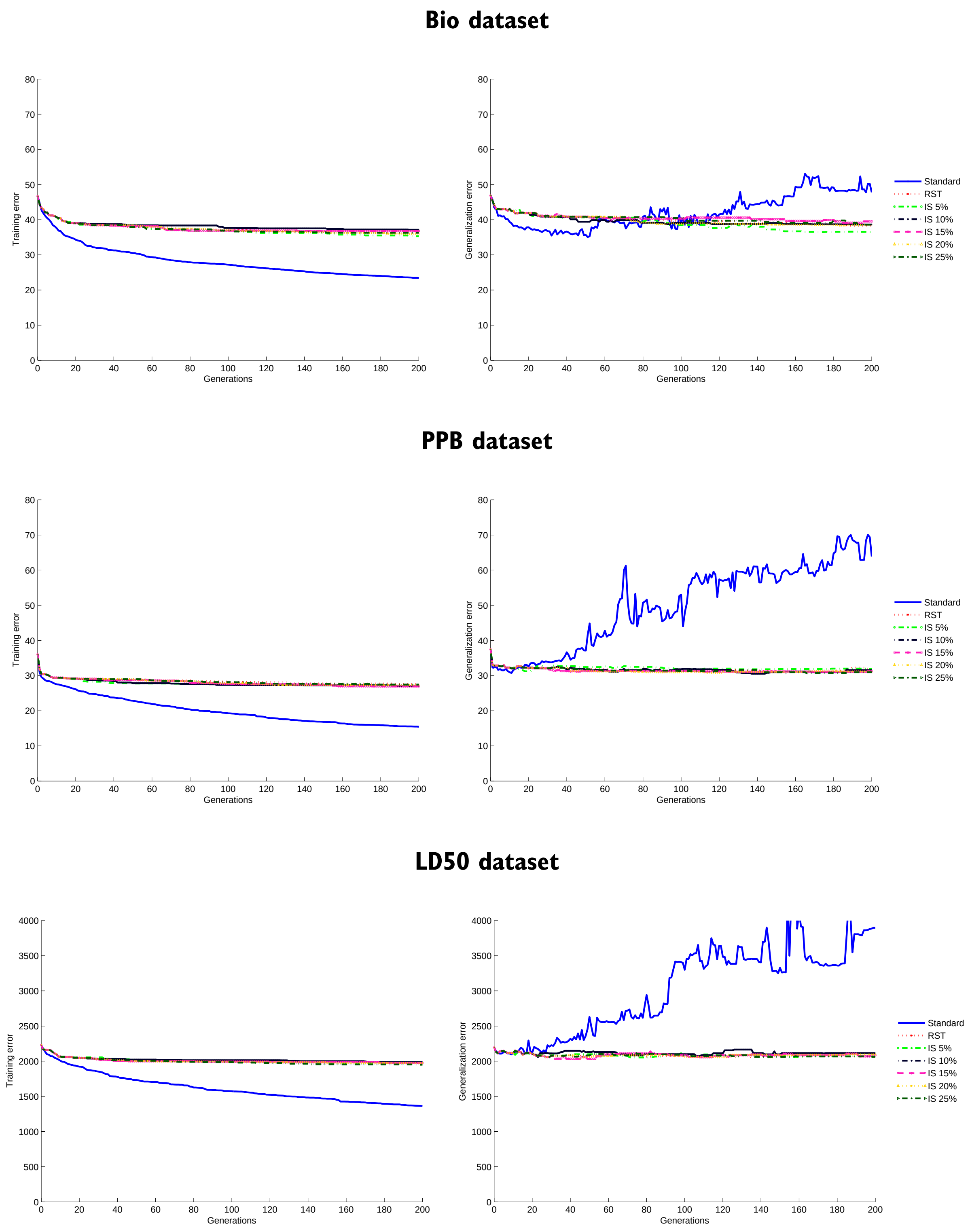


Figure 4.9: Training and generalization errors evolution plots for Standard GP, RST, and IS with different parameters

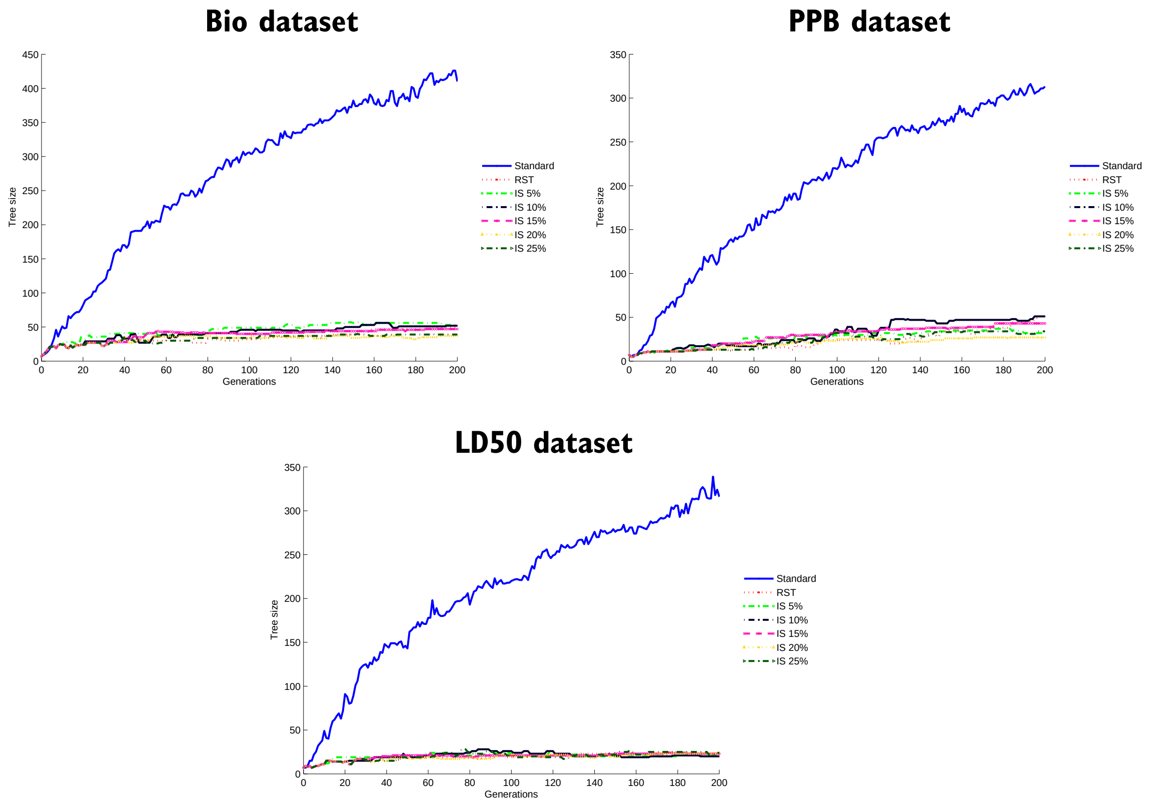


Figure 4.10: Tree size evolution plots for Standard GP, RST, and IS with different parameters

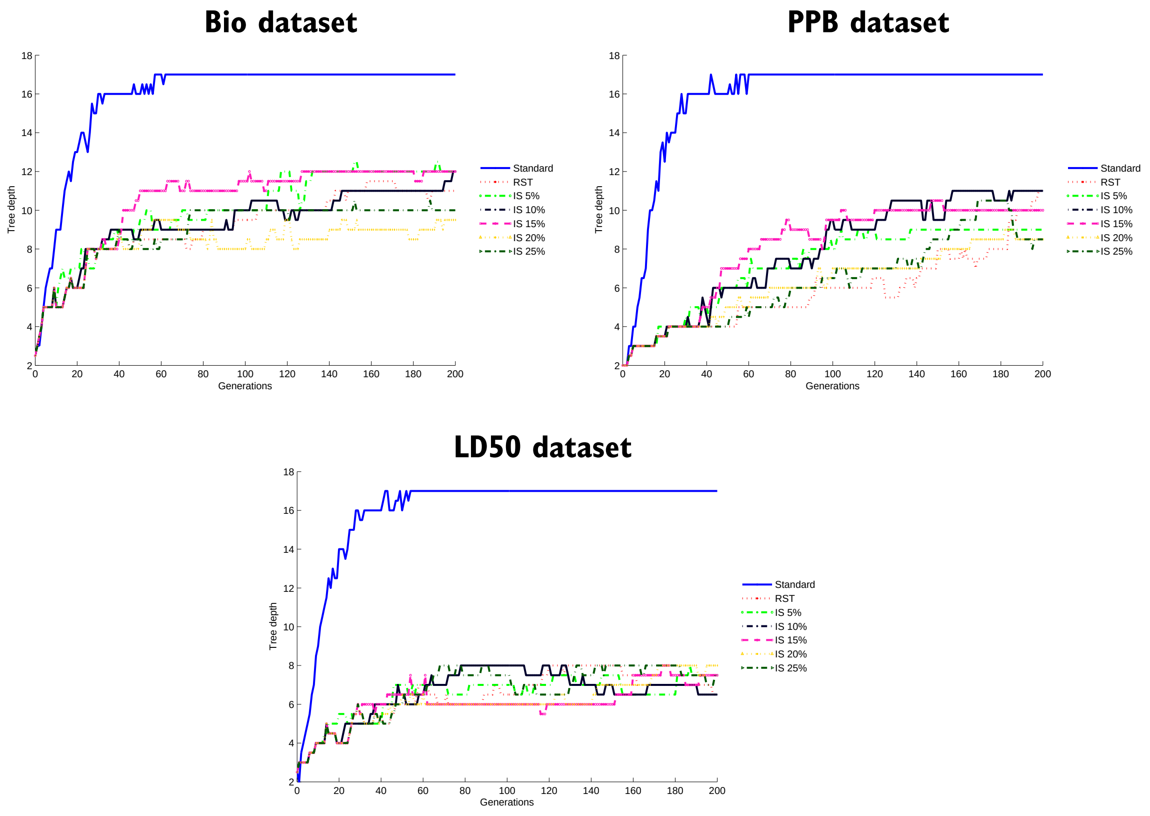


Figure 4.11: Tree depth evolution plots for Standard GP, RST, and IS with different parameters

4.3 Random Interleaved Sampling

4.3.1 Method Description and Results

The Random Interleaved (RI) approach is based on a probabilistic choice of the generations where a single training instance is used. At each generation the number of training instances to use is decided by considering a probability (given as a parameter) of using a single training instance. In the generations where a single training instance is not used, all of the training data is used instead. The parameter of RI is represented as a percentage. The values tested for this parameter are: 99%, 95%, 75%, 50%, 25%, and 5%. Notice that using 100% as a parameter is equivalent to the RST approach of always using a single training instance. Notice as well that using 0% as a parameter is equivalent to the Standard GP approach of always using all the training data.

Figure 4.12 presents the training and generalization errors evolution plots for Standard GP, RST, and RI with the different parameters tested. The corresponding overfitting plots are presented in figure 4.13. The RI variants exhibit the expected behavior with respect to their parametrization. The closer the parameter is to 100%, the closer the variant behaves to RST. Conversely, the closer the parameter is to 0%, the closer the variant behaves to Standard GP. In terms of generalization, the RI configurations with the parameter set to 50% or above usually achieve competitive generalizations. Figure 4.14 shows the boxplots with the generalization errors achieved in the last generation of each variant. Particularly interesting are the results of RI 50% and 75% in the Bio dataset. These variants are able to achieve a better generalization than Standard GP with statistically significant differences (p-values: RI 50% $\mathbf{6.161 \times 10^{-5}}$, and RI 75% $\mathbf{1.905 \times 10^{-3}}$). As previously seen, RST was not able to do so. Besides this advantage, RI 50% and 75% also increase the training data learning rate in comparison with RST. In the PPB dataset, RI configurations with the parameter set to 50% or above significantly improve the generalization in comparison with Standard GP: 50% (p-value $\mathbf{2.320 \times 10^{-4}}$), 75% ($p$-value $\mathbf{1.024 \times 10^{-7}}$), 95% ($p$-value $\mathbf{1.774 \times 10^{-9}}$), and 99% ($p$-value $\mathbf{1.800 \times 10^{-7}}$). In the LD50 dataset, generalization improvements over Standard GP occur with statistical significance for the RI configurations with the parameter set to 75% or above: 75% (p-value $\mathbf{5.434 \times 10^{-5}}$), 95% ($p$-value $\mathbf{2.290 \times 10^{-8}}$), and 99% ($p$-value $\mathbf{1.024 \times 10^{-7}}$). These results reinforce the notion that, in order to avoid overfitting, a preference has to be given toward using only a single training instance at each generation.

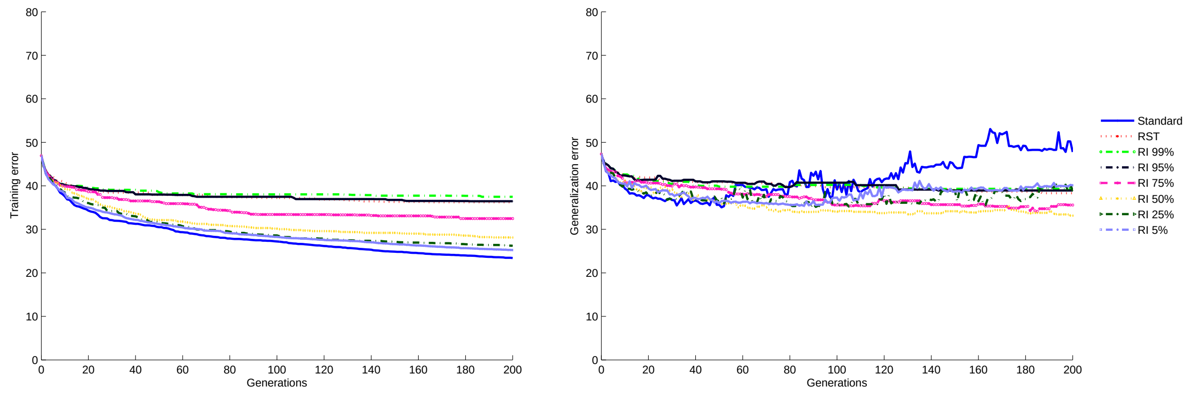
In terms of tree size (figure 4.15), the RI configurations with the parameter set to 50% or above always produce smaller individuals than Standard GP. The advantages in terms of tree depth (figure 4.16) are particularly clear for RI 75% and above.

4.3.2 Normalized Comparison

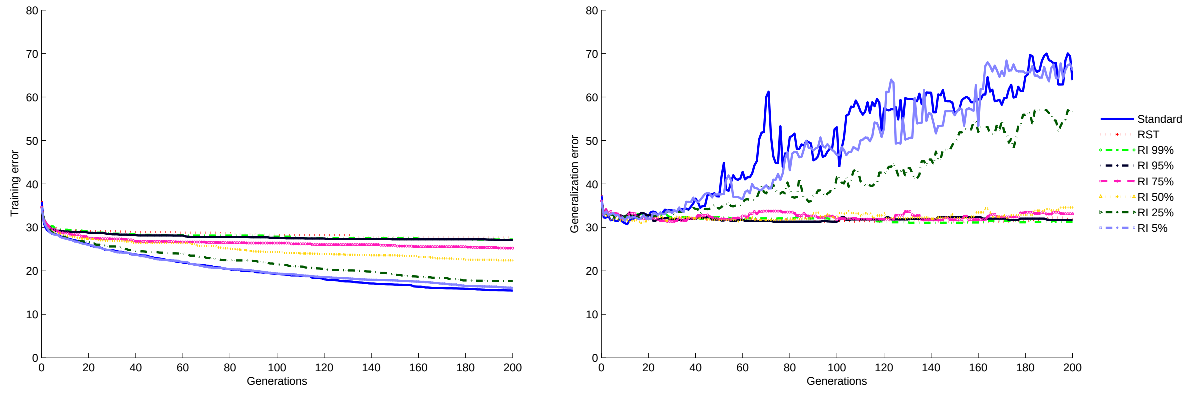
Despite the advantages presented by the approaches studied in this chapter, a possible source of confusion still needs to be addressed. As it stands, it is not clear if the proposed approaches simply worked by delaying overfitting. An argument could be made that, by providing fewer training instances in total number during the run, the search algorithm would simply stop (given the same number of total generations) before overfitting starts. To address this issue, an experiment is performed in order to extend the runs of the proposed approaches so that, at the end of the runs, these approaches would be presented in total with at least the same number of training instances as Standard GP. In other words, a normalized comparison is performed in terms of total number of training instances seen. Given their positive results, RI 50% and RI 75% are used in this experiment. Extending the number of generations to 400 in RI 50%, means that on average the search method would use all of the training data in 200 generations. This is exactly the number of generations used in Standard GP. This entails that running RI 50% for 400 generations guarantees on average the usage of at least the same number of total training instances. Although not crucial for the comparison being made, the number is actually a little bit higher as the remaining 200 generations use a single training instance. Following the same reasoning, extending the number of generations to 800 in RI 75%, means that on average the search method would use all of the training data in the same 200 generations. With these experimental settings, RI 50% and 75% will use on average at least the total number of training instances used in Standard GP. The RI variants with an extended number of generations are named Normalized RI (NRI). Figure 4.17 shows the boxplots with the generalization errors achieved in the last generation of each variant.

In the total of 6 comparisons between the NRI and the RI variants, only once does the normalizing procedure result in a considerably weaker generalization (NRI 50% in the PPB dataset). On the other hand, in another case (in the LD50 dataset) the NRI variant is able to surpass the Standard GP outcome where the corresponding RI variant could

Bio dataset



PPB dataset



LD50 dataset

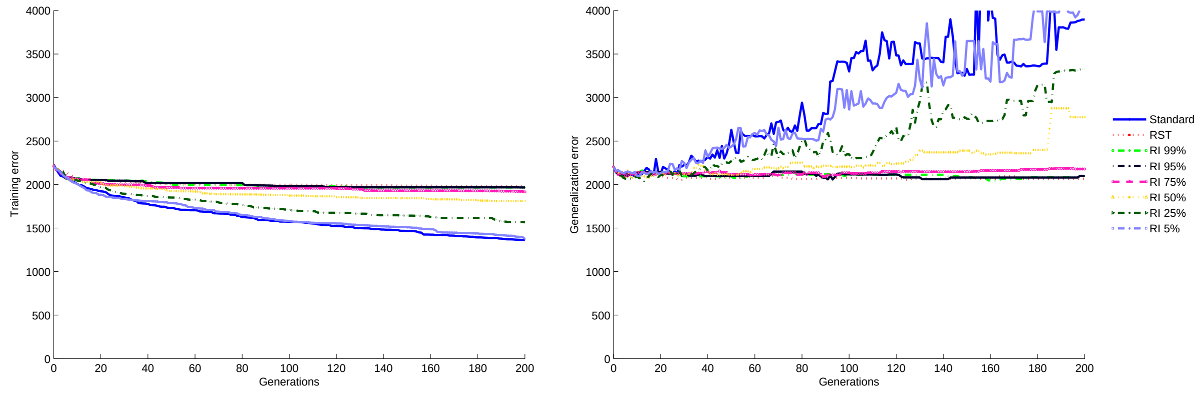


Figure 4.12: Training and generalization errors evolution plots for Standard GP, RST, and RI with different parameters

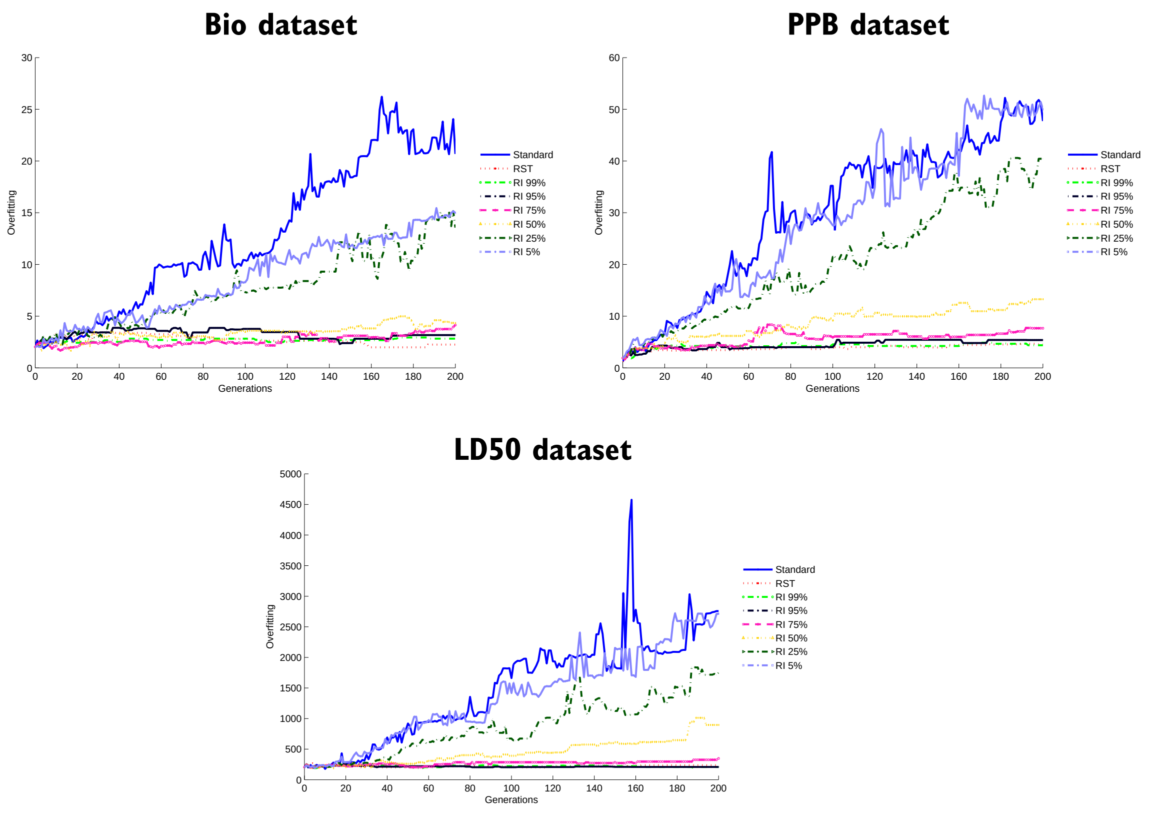


Figure 4.13: Overfitting evolution plots for Standard GP, RST, and RI with different parameters

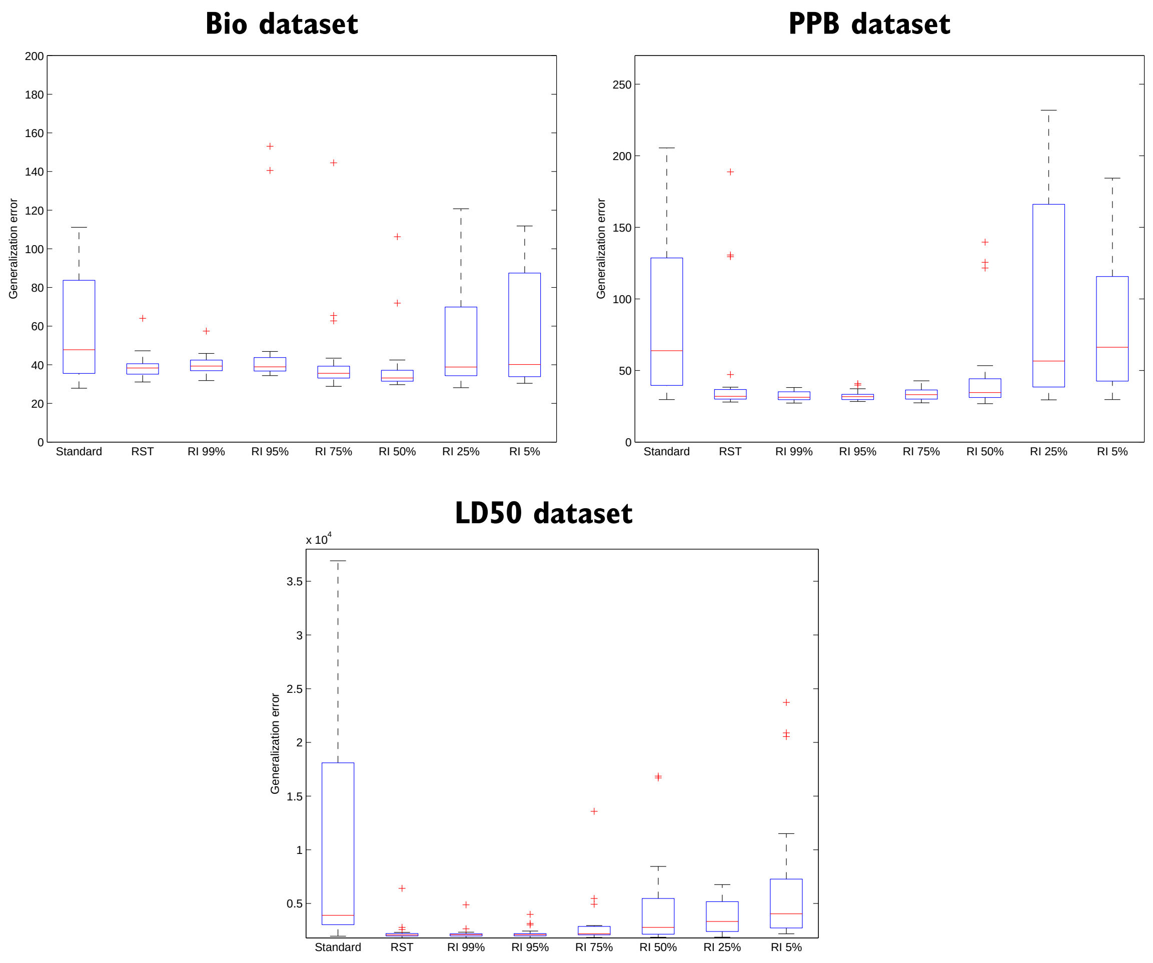


Figure 4.14: Generalization error boxplots for Standard GP, RST, and RI with different parameters

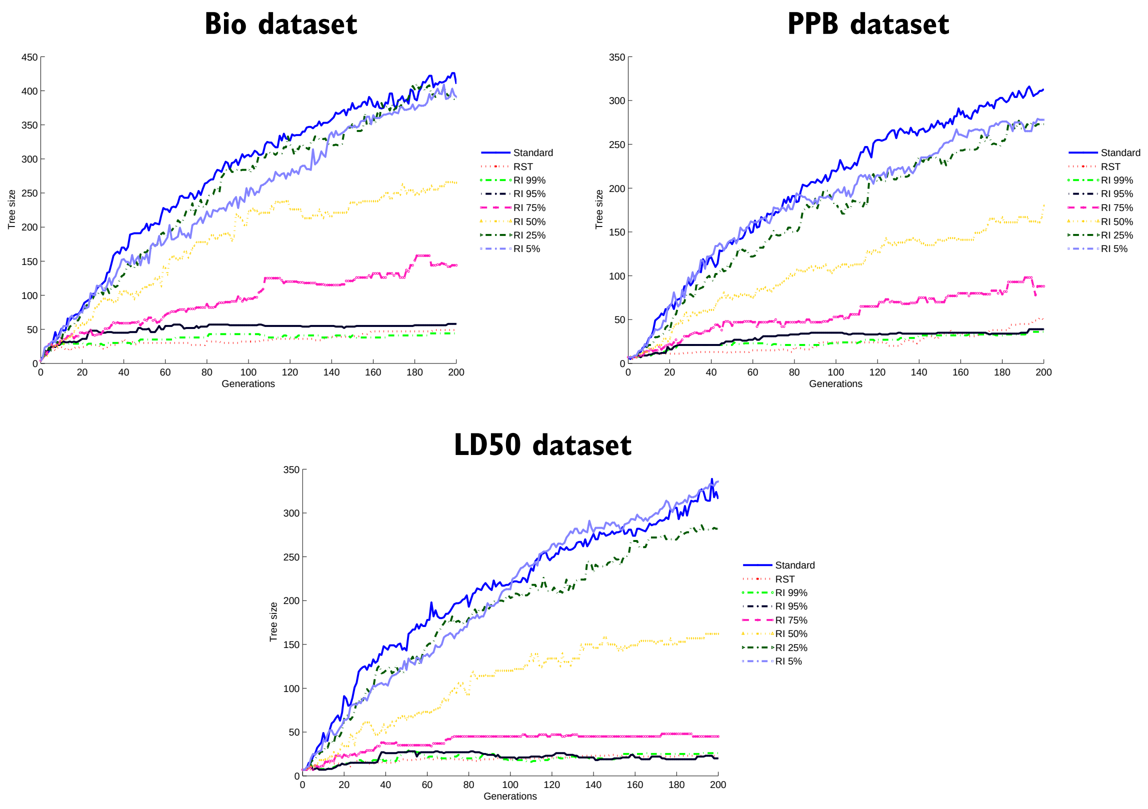


Figure 4.15: Tree size evolution plots for Standard GP, RST, and RI with different parameters

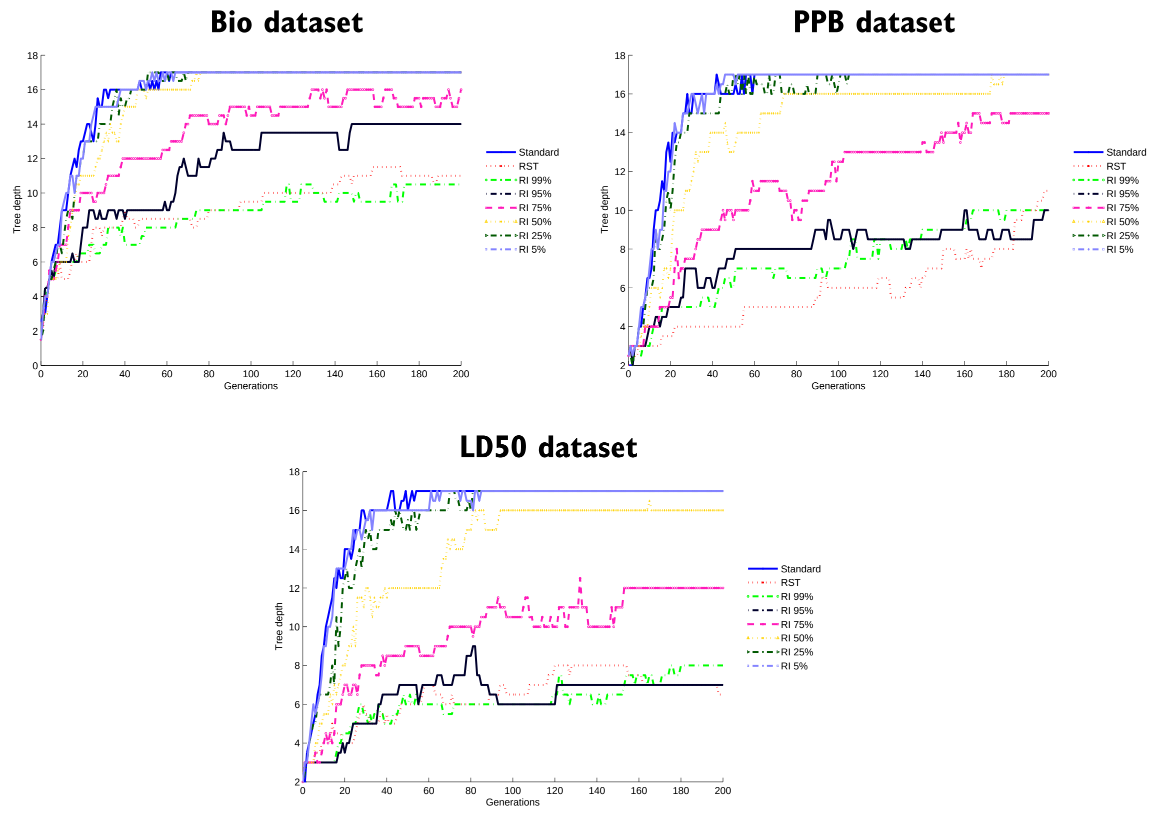


Figure 4.16: Tree depth evolution plots for Standard GP, RST, and RI with different parameters

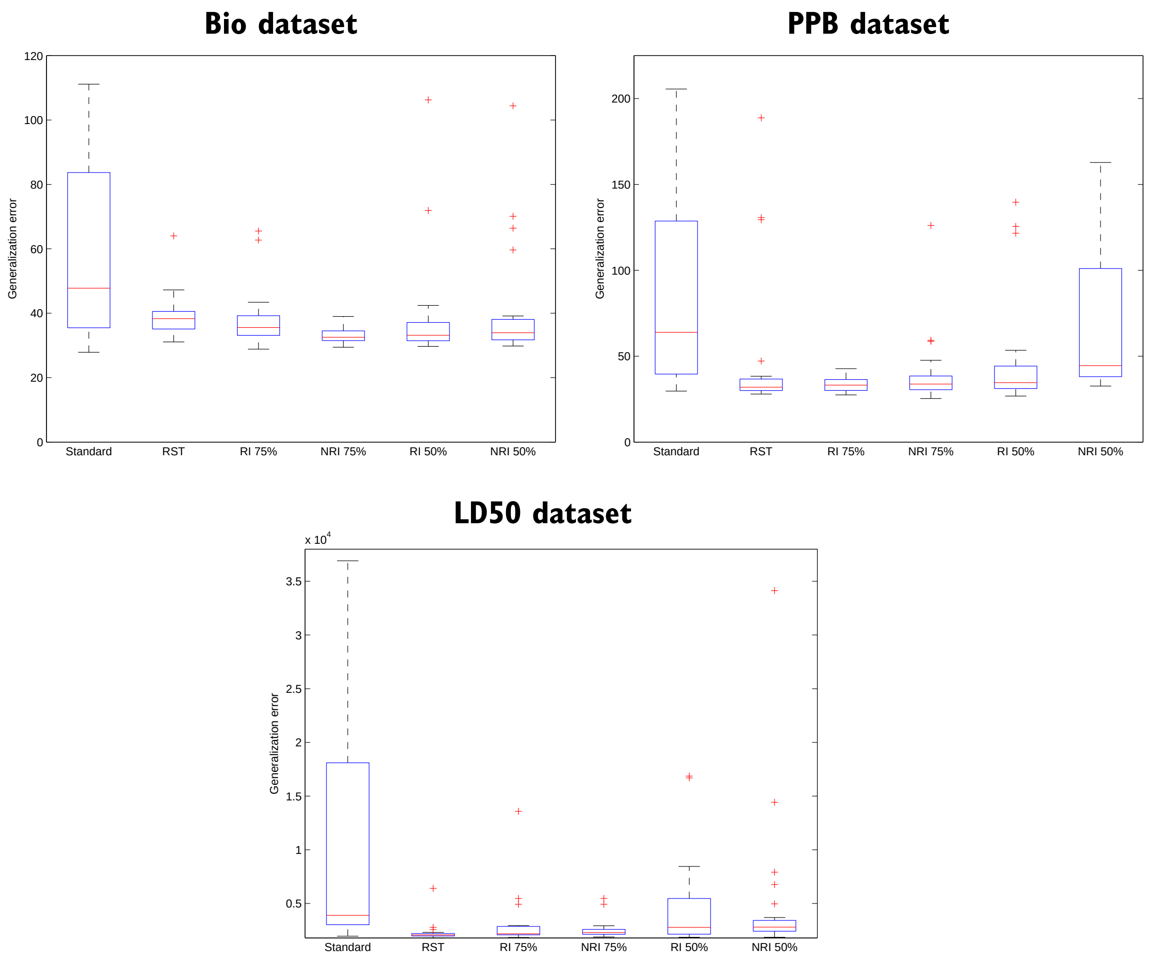


Figure 4.17: Generalization error boxplots for Standard GP, RST, RI 50% and 75%, and NRI 50% and 75%

not (p-value NRI 50% 4.128×10^{-3}). In the remaining 4 comparisons, the generalization superiority over Standard GP remains in the normalized variants: NRI 75% (p-values: Bio 1.067×10^{-6}, PPB 1.067×10^{-6}, and LD50 8.401×10^{-5}), and NRI 50% (p-value Bio 3.666×10^{-4}).

With these results, the possibility that the RI approach is simply delaying overfitting can be excluded. This approach is in fact avoiding overfitting by interleaving the usage of all the training data with the usage of a single training instance.

5 On the Generalization Ability of Geometric Semantic Genetic Programming

This chapter explores the generalization ability of Geometric Semantic Genetic Programming (GSGP), a recently proposed form of GP. Section 5.1 introduces GSGP. Section 5.2 provides an analysis of the geometric semantic operators. Section 5.3 explores the consequences of computing optimal weights. Section 5.4 shows how alignments in the error space can be achieved.

5.1 Geometric Semantic Genetic Programming

Geometric Semantic Genetic Programming (GSGP) is a novel form of GP recently proposed by Moraglio et al. (2012). Particularly, GSGP encompasses a set of new variation operators that share some specific characteristics. All GSGP formulations are valid under two assumptions. The first assumption is that the task at hand is a supervised learning task, i.e., given a set of data with known targets, the goal is to build an individual (or model) that learns the underlying patterns in the data. The second assumption is that the error of an individual is computed as a distance to the known targets. This distance must be formally interpreted as a metric. GSGP derives its name from the fact that its operators follow some particular geometric properties (Moraglio, 2007) over the semantic space. In this context, the semantics of an individual are defined as the outputs of that individual over a set of data instances. In GSGP, each individual is seen as a point in the semantic space. Consequently, a geometric semantic crossover is an operator that always creates an offspring with semantics between the semantics of both parents. In other words, given any data instance, the output of the generated offspring must be between the outputs of the parents for that same data instance. A geometric semantic mutation is an operator that always creates an offspring with semantics that are within a given distance of the semantics of its parent. Depending on the version of the geometric semantic mutation, this distance may be possible to specify by a mutation step. The most interesting property of these geometric semantic operators is that they induce an unimodal fitness landscape with a constant slope for any supervised learning problem. Formally, this fitness landscape is a metric cone. This implies that there are no local optima, i.e., with the exception of the global optimum, every point in the search space has at least one neighbor with better fitness, and that neighbor is reachable through the application of the variation operators. As this type of landscape eliminates the issue of the local optima, it is potentially much more favorable in terms of search efficiency. Moraglio et al. (2012) specified geometric semantic operators for three domains: boolean, arithmetic, and program (conditional rules). Since this work is based on real-valued semantics, the geometric semantic operators presented here are the ones from the arithmetic domain.

Definition 1 Geometric Semantic Crossover: Given two parent functions $T_1, T_2 : \mathbb{R}^n \rightarrow \mathbb{R}$, the geometric semantic crossover returns the real function $T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$, where T_R is a random real function whose output values range in the interval $[0, 1]$.

Another version of this crossover can be defined by replacing the random real function (T_R) by a random real constant, as long as its value is also in the interval $[0, 1]$ (Moraglio et al., 2012).

Definition 2 Geometric Semantic Mutation: Given a parent function $T : \mathbb{R}^n \rightarrow \mathbb{R}$, the geometric semantic mutation with mutation step ms returns the real function $T_M = T + ms \cdot (T_{R1} - T_{R2})$, where T_{R1} and T_{R2} are random real functions.

The main drawback of the geometric semantic operators is that they always produce offspring bigger that their parents. Particularly for the crossover, every offspring is bigger than its parents combined. This leads to an exponential individual growth. For the mutation operator, the individual growth is linear. This continuous individual growth produced by the geometric semantic operators renders GSGP hard to use in practice, particularly in real-world multidimensional datasets. To counteract this growth, Moraglio et al. (2012) proposed a simplification step after each operator application. This simplification approach was effective for the artificial problems tested, but it remains to be shown if it is effective in real-world multidimensional datasets. Another open question raised from the original GSGP work, is the issue of generalization. Moraglio et al. (2012) did not measure the performance of the individuals in unseen data. Therefore, the generalization ability of the individuals produced by GSGP was unknown.

Vanneschi et al. (2013) tackled the individual growth issue by reimplementing the operators to compute the semantics of the offspring without explicitly building their syntax. This implementation does not reduce the growth of the individuals, but it allows for an estimation of the GSGP performance without dealing with the issues that arise with the actual creation of the individuals. This allowed Vanneschi et al. (2013) to use GSGP for the first time in real-world multidimensional datasets. Results showed competitive performance both in training and unseen data. The arguments presented for the competitive unseen data performance are described and discussed in the next section. However, the implementation of the mutation operator of Vanneschi et al. (2013) had a small deviation from the original definition (Moraglio et al., 2012). Their implementation imposed that the random subtrees generated (T_{R1} and T_{R2}), always had a logistic function (5.1) as their root node.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5.1}$$

This implies that the output of each random subtree ranges in the interval $[0, 1]$ and that, consequently, the output resulting from subtracting these random subtrees ranges in the interval $[-1, 1]$. As the mutation operator applies a mutation step (ms), the final output added to the parent always ranges in the interval $[-ms, ms]$. Looking back at the original definition of the geometric semantic mutation (Moraglio et al., 2012), there is no defined range for the outputs of the random subtrees. It should be noted that this small implementation deviation is still valid under the geometric semantic framework. This deviation was not explicit in their work but it can be confirmed in the mutation implementation provided by the same authors in their GSGP library (Castelli et al., 2014). For clarification purposes, the original mutation definition is referred to as Unbounded

Mutation (UM), and the alternative mutation implementation is referred to as Bounded Mutation (BM). In the end, BM applies a structural bound on the perturbation applied to the parent. This bound holds independently of the data (training or unseen). In the next section, it is shown that these different mutations lead to considerably different outcomes in terms of generalization. The reasons behind these behaviors are also explored.

5.2 An Analysis of the Geometric Semantic Operators

To provide a fair comparison between unbounded and bounded mutation, the experimental setup is similar to the one of Vanneschi et al. (2013), where the bounded mutation was first tested. The experimental parameters that are different from the baseline parameters are provided in table 5.1. Experiments are run for 500 generations because that is where the statistical comparisons were made in the mentioned work. Vanneschi et al. (2013) used the Bio and PPB datasets in their experiments. Here, besides these two, the LD50 dataset is also used as in the previous chapter.

Table 5.1: GSGP and Standard GP parameters used in the experiments

Parameter	**Value**
Population size	100
Generations	500
Data partition	Training 70% - Unseen 30%
GSGP crossover	Probability 0.5
GSGP mutation	Probability 0.5
Terminal set	Input variables, no constants
Parent selection	Tournament selection of size 4
Maximum tree depth	None

5.2.1 Initial Exploration

As previously mentioned, the bounded mutation imposes that the semantic variation added to the parent always ranges in the interval $[-ms, ms]$. On the other hand, while the unbounded mutation is still influenced by the mutation step, it is not possible to define a priori a bound for the semantic variation that is going to be imposed to the parent.

This means that even for the same mutation step, both mutation operators may have considerable differences in the amount of semantic variation that is imposed to the parent. The following set of experiments are conducted in order to exclude the possibility that the differences in terms of generalization may result from the different amount of semantic variation that each operator imposes.

Figures 5.1 to 5.3 show the evolution plots for GSGP with bounded (BM) and unbounded (UM) mutation, with several mutation steps (MS in the figures), respectively for the Bio, PPB, and LD50 datasets. The main result across the three datasets is that GSGP UM generally overfits the training data, while GSGP BM presents a smoothing effect on the generalization error that eventually leads to a good generalization. These behaviors are consistent regardless of the mutation step selected. The same conclusions hold even if a small mutation step is selected for GSGP UM (e.g., 0.1), and a high mutation step is selected for GSGP BM (e.g., 100). These results show that the different amount of semantic variation that each operator imposes does not explain the different behaviors in terms of generalization. This is particularly clear when GSGP UM overfits even when small mutation steps are used. It seems that structurally bounding the semantic variation that can be applied to the parent is indeed, and by itself, the reason for the smooth effect on the generalization ability. These results show that the generalization ability can not be simply justified by the properties of the geometric semantic operators. If this was the case, then any valid formulation of the geometric semantic mutation would achieve similar generalization outcomes. On the contrary, these results show that different formulations of the geometric semantic mutation, do indeed lead to considerably different results. In terms of training data performance, it is clear that the mutation step is very influential. Since higher mutation steps allow for bigger semantics adjustments, the higher the mutation step the bigger the potential training error decrease. This can be clearly seen at least in the beginning of the runs. This is also why GSGP UM fits the training data faster, since the semantic variation can be much higher than GSGP BM with the same mutation step. On the other hand, given a fixed mutation step, there is always a point in the evolution where the mutation step becomes too high to be effective. As an individual gets closer to the target, the level of semantic adjustments needed to keep fitting the data become smaller. Consequently, and in order to be effective, the mutation step should also decrease as the distance to the target decreases. A dynamic mutation step adaptation performed throughout the run can guarantee that the mutation step is never too high in any given situation. This approach is explored in the next section. For the remaining experiments presented in this chapter, the mutation step for the Bio and PPB datasets is set to 1, and the mutation step for the LD50 dataset is set to 10. These values were selected because they result in a good performance in terms of learning the training data, for both mutation formulations.

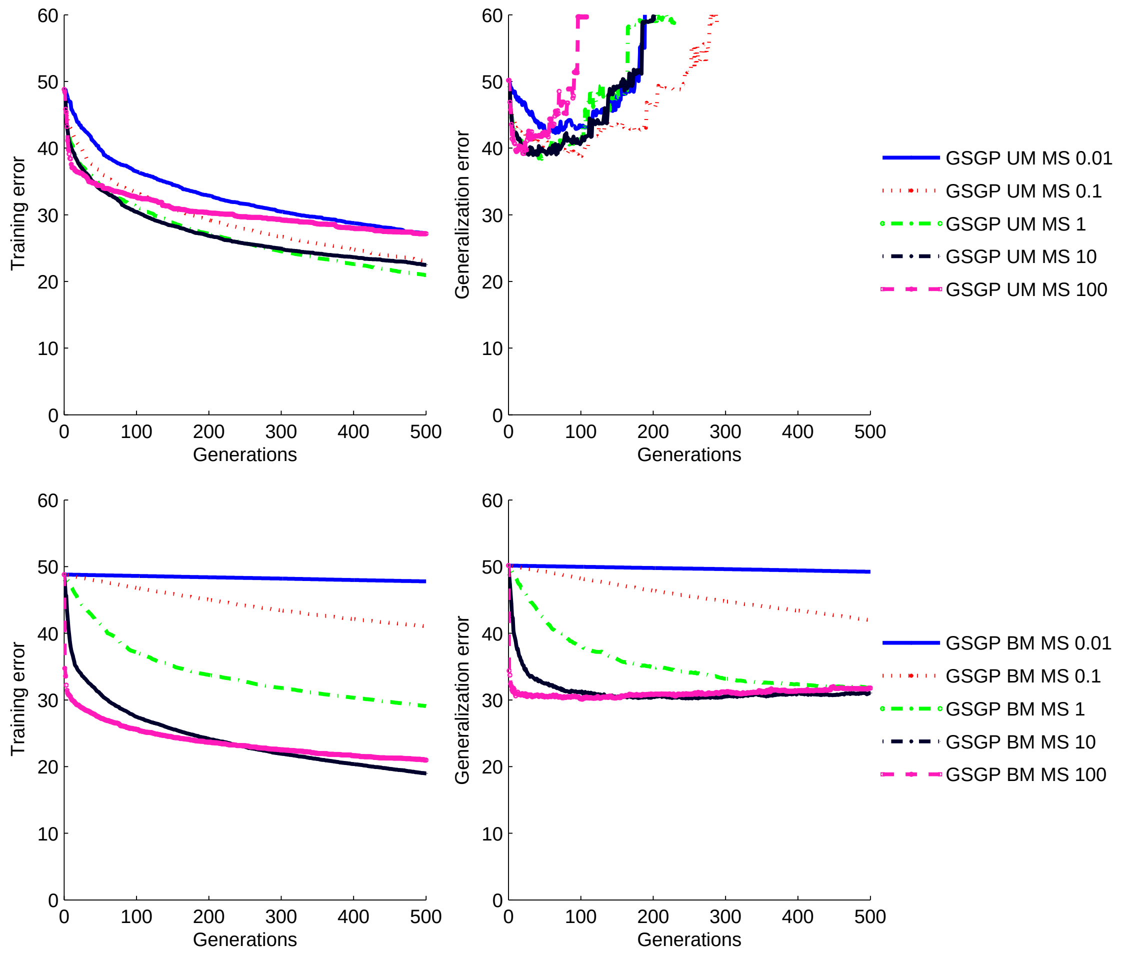


Figure 5.1: Training and generalization errors evolution plots for GSGP UM (top) and BM (bottom), with several mutation steps in the Bio dataset

Another effect that needs to be excluded is the possibility that the bounding function selected (i.e., the logistic function) for the random subtrees (T_{R1} and T_{R2}) is particularly helpful in these datasets, and consequently, this is what explains the different results. To assess this possibility, two different bounding functions are tested. The first function is a

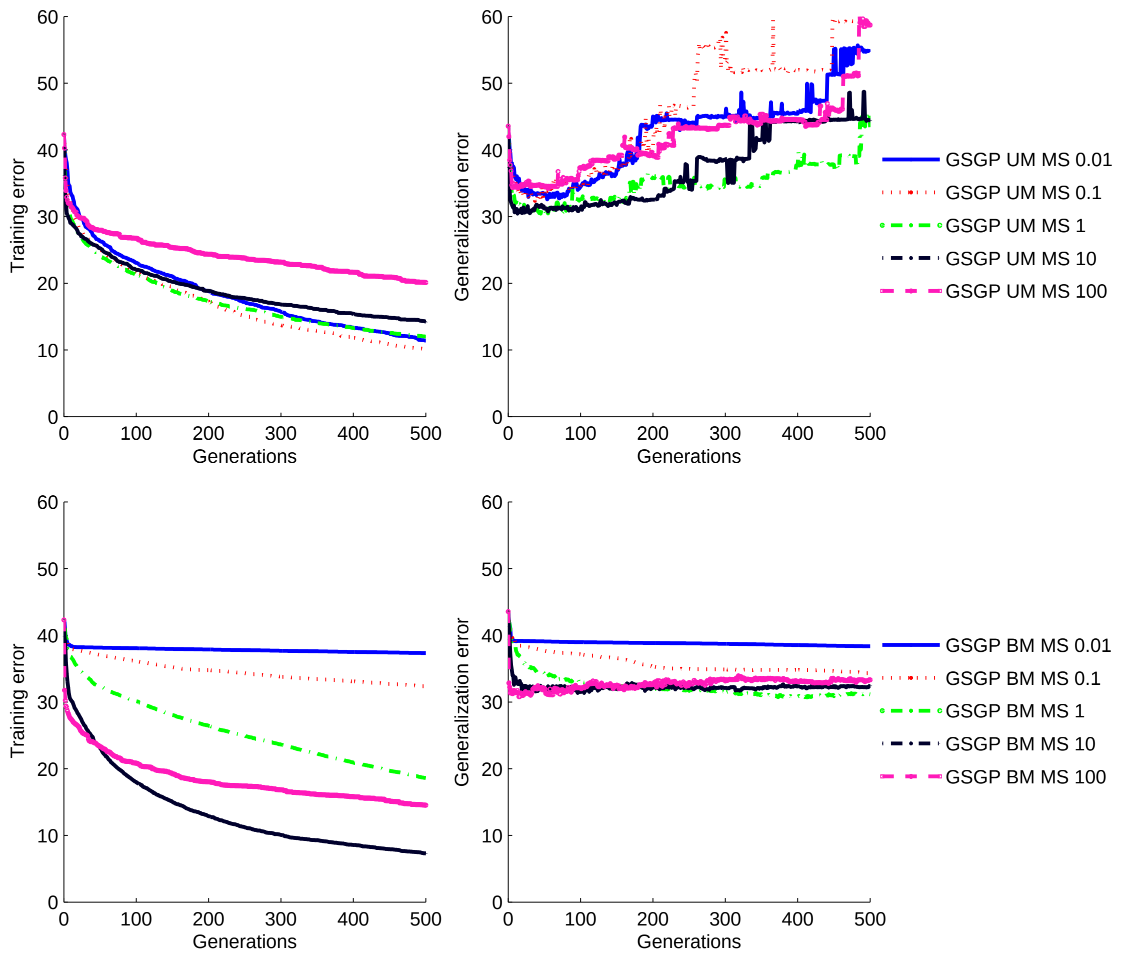


Figure 5.2: Training and generalization errors evolution plots for GSGP UM (top) and BM (bottom), with several mutation steps in the PPB dataset

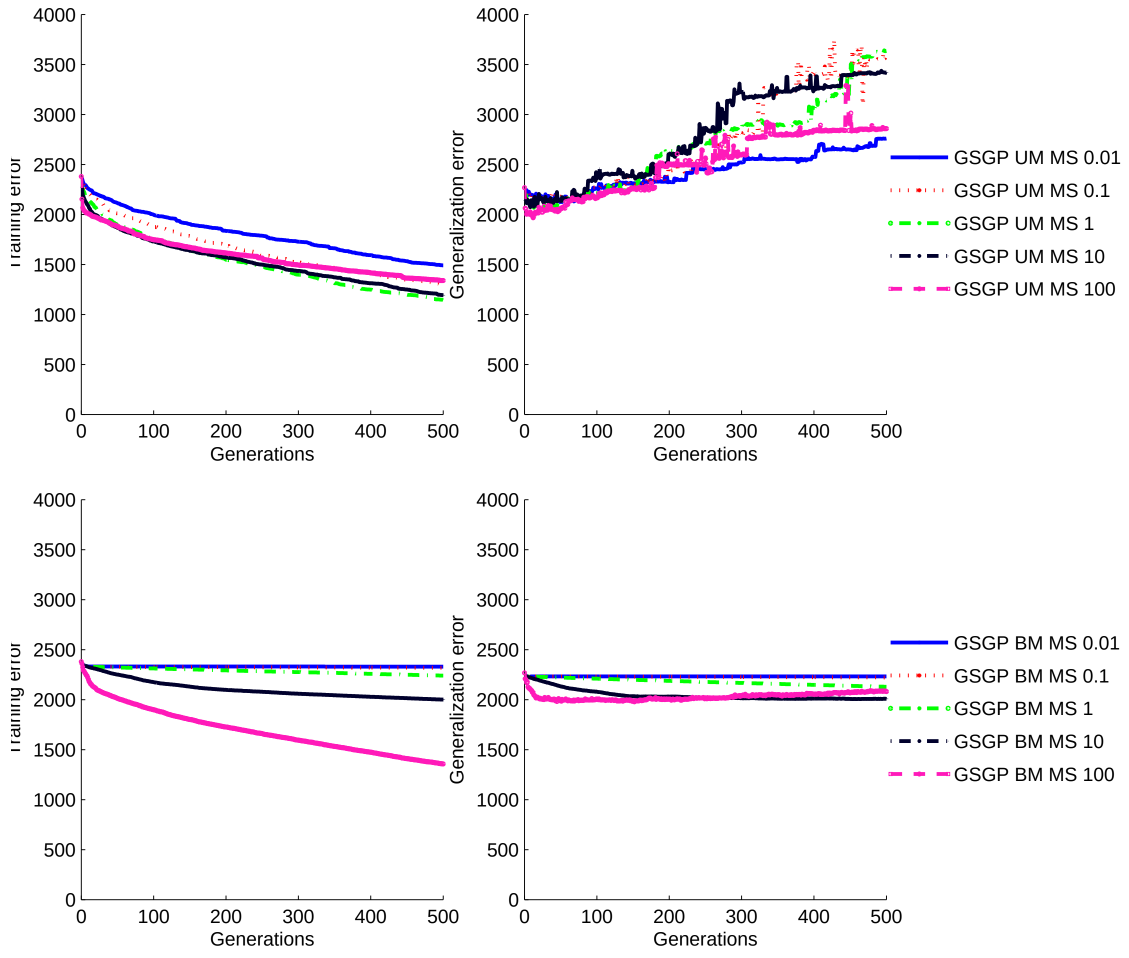


Figure 5.3: Training and generalization errors evolution plots for GSGP UM (top) and BM (bottom), with several mutation steps in the LD50 dataset

linear bound defined as:

$$f(x) = \begin{cases} 0 & ,x < 0 \\ 1 & ,x > 1 \\ x & \text{, otherwise} \end{cases} \tag{5.2}$$

The second function is a non-linear bound that uses the sine function. It is defined as:

$$f(x) = \frac{\sin(x)}{2} + 0.5 \tag{5.3}$$

To ensure a fair comparison all three functions bound the outputs in the same interval: $[0, 1]$. Figure 5.4 shows the evolution plots for GSGP BM with the different bounding functions presented, for all datasets tested. Results show that all bounding functions perform similarly in terms of training and generalization errors. The only exception occurs in the PPB dataset, where the Sine bound helps to fit the training data faster than when using the other bounding functions. This, however, does not translate into a better generalization. With these results, it can be excluded the possibility of the logistic function being particularly helpful in these datasets. As it stands, the good generalization ability of GSGP BM is explained by the structural bound of the semantic variation applied to the parent at each mutation operation. This seems to hold independently of the particular bounding function selected. Since the selection of the bounding function did not considerably influence the results, for the remaining experiments in this chapter the logistic function will be kept as the bounding function as in the work of Vanneschi et al. (2013).

5.2.2 Extended Comparison

The next set of experiments are aimed at assessing the performance of different GSGP configurations. Standard GP is used as a baseline for comparison. Since GSGP does not impose a limit on the depth of each individual, the Standard GP version used here also does not apply a depth limit. This allows a fair comparison to be made between both GP systems. A variant of Standard GP with the common depth limit of 17 is also tested. This variant is named Standard DL 17. The Semantic Stochastic Hill Climber (SSHC) (Moraglio et al., 2012) is also used as a baseline for comparison. SSHC uses only the geometric semantic mutation to explore the search space. At each generation, 100

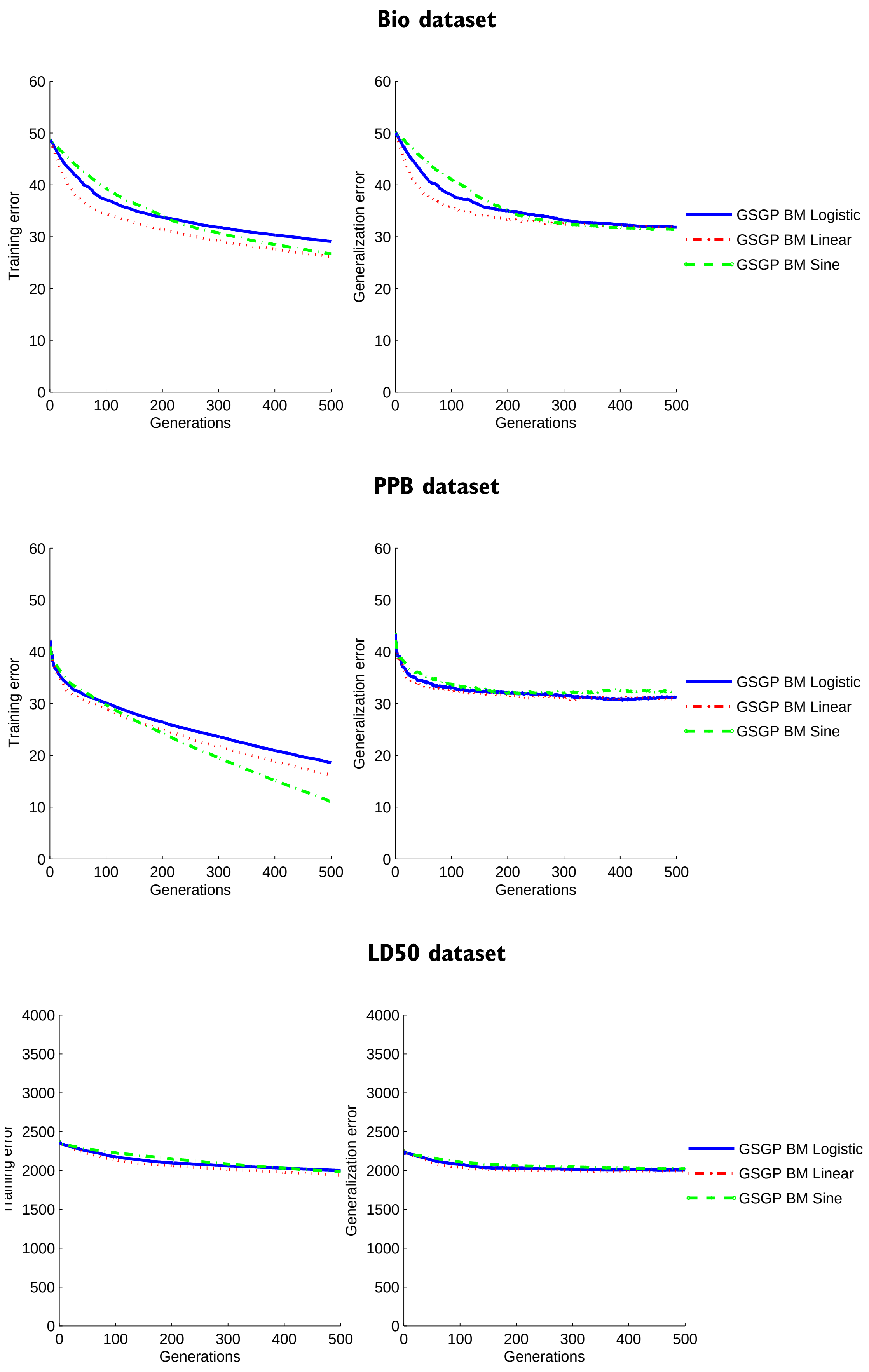


Figure 5.4: Training and generalization errors evolution plots for GSGP BM with different bounding functions

neighbors (offspring) are created. The best individual selected from the previous best and the neighbors survives to the next generation. Notice that the number of neighbors generated in SSHC is the same as the population size of GSGP and Standard GP. This guarantees that at the end of each run, all methods evaluated the same total number of individuals. GSGP without crossover (GSGP NC) is also tested. As this section studies the effects of unbounded and bounded mutations (UM and BM respectively), all methods that use the geometric semantic mutation are tested with both mutations. Therefore, the geometric semantic variants tested are: GSGP UM and BM; GSGP UM NC and BM NC; and SSHC UM and BM.

Figure 5.5 shows the training and generalization errors evolution plots for the datasets tested, and figure 5.6 shows the corresponding overfitting evolution plots. The overall trend across all datasets is that Standard GP and the UM variants (GSGP UM, GSGP UM NC, and SSHC UM) overfit the training data, while the BM variants (GSGP BM, GSGP BM NC, and SSHC BM) generalize well. The discrepancy between the generalization ability of the UM and BM variants is discussed in the next subsection. On the other hand, the UM variants are generally the most efficient variants in fitting the training data.

Statistically it is confirmed that GSGP BM generalizes better than GSGP UM in all datasets (p-values: Bio $\mathbf{1.943 \times 10^{-9}}$; PPB $\mathbf{4.915 \times 10^{-6}}$; LD50 $\mathbf{1.537 \times 10^{-10}}$). In terms of training data performance, the superiority of GSGP UM is also confirmed in all datasets (p-values: Bio $\mathbf{2.872 \times 10^{-11}}$; PPB $\mathbf{2.872 \times 10^{-11}}$; LD50 $\mathbf{2.872 \times 10^{-11}}$). In the comparisons against Standard GP, it was confirmed that GSGP BM generalizes better in all datasets (p-values: Bio $\mathbf{1.794 \times 10^{-6}}$; PPB $\mathbf{5.121 \times 10^{-4}}$; LD50 $\mathbf{3.657 \times 10^{-9}}$). Similar conclusions were presented by Vanneschi et al. (2013) for the Bio and PPB datasets. In terms of training data performance, Standard GP is superior to GSGP BM in the Bio (p-value $\mathbf{5.434 \times 10^{-5}}$) and LD50 ($p$-value $\mathbf{8.563 \times 10^{-11}}$) datasets, and no statistically significant differences were found in the PPB dataset. However, and as seen previously, the mutation step greatly affects the training data performance of GSGP (in both mutation versions). If the objective was to fit the training data faster, then a bigger mutation step would translate into a superior GSGP performance, as seen in figures 5.1, 5.2, and 5.3. Against GSGP UM, Standard GP is superior in terms of generalization error in the Bio dataset (p-value $\mathbf{9.273 \times 10^{-4}}$), while no statistically significant differences were found in the PPB and LD50 datasets. The statistically significant advantage in the Bio dataset is however not particularly relevant as both methods clearly overfit. In terms of learning the training data, GSGP UM is superior to Standard GP in all datasets (p-values: Bio 2.872×10^{-11}; PPB 2.872×10^{-11}; LD50 6.373×10^{-11}).

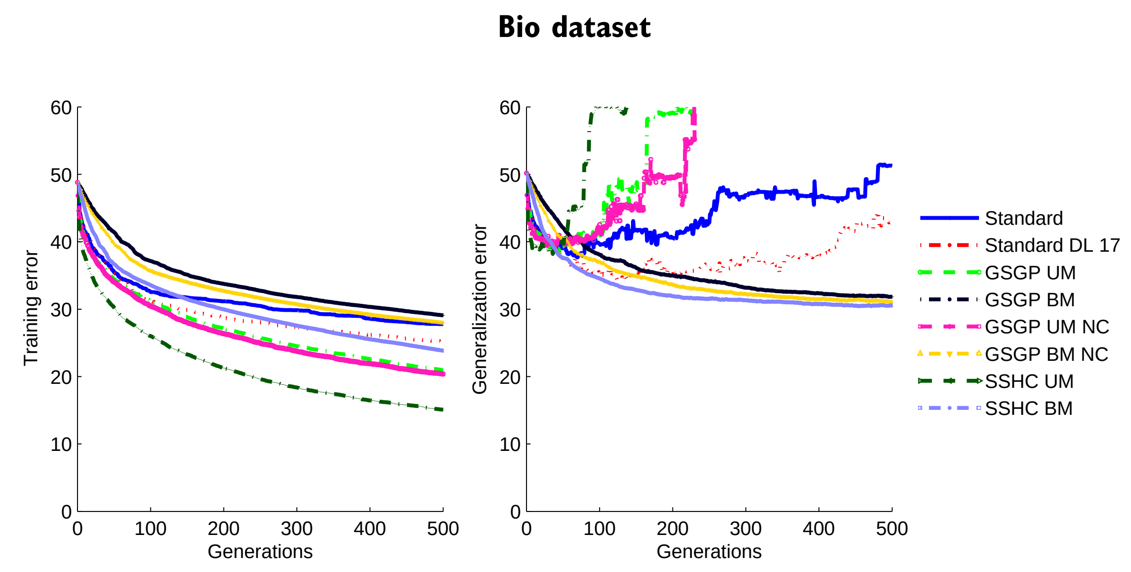
The evolution plots also show that the SSHC variants consistently learn faster than the GSGP and GSGP NC variants with the same mutation operator. This should be expected as the semantic space has no local optima and, consequently, the search can be focused around the best individual in the population. This leads to a faster decrease in the training error and, in the case of SSHC BM, to a faster generalization to unseen data. Regarding the usage of a depth limit in Standard GP, both variants (Standard and Standard DL 17) presented relatively similar behaviors. The only considerably different outcome occurs in the PPB dataset, where Standard DL 17 overfits more the training data. In terms of learning the training data, the imposition of a depth limit did not affect the search effectiveness. Figure 5.7 shows the boxplots with the generalization errors achieved in the last generation of each method. In order to properly visualize the results, some outliers are not shown for the Standard GP and the UM variants. The main conclusion from these boxplots is the robustness of the BM variants. These variants achieve consistent values in all datasets. On the other hand, the Standard GP and the UM variants present considerable variation in the generalization values achieved, i.e., different runs of the same method can behave very differently in terms of generalization. The UM variants achieved particularly high generalization errors, and therefore, had to be presented in different figures with higher generalization error limits.

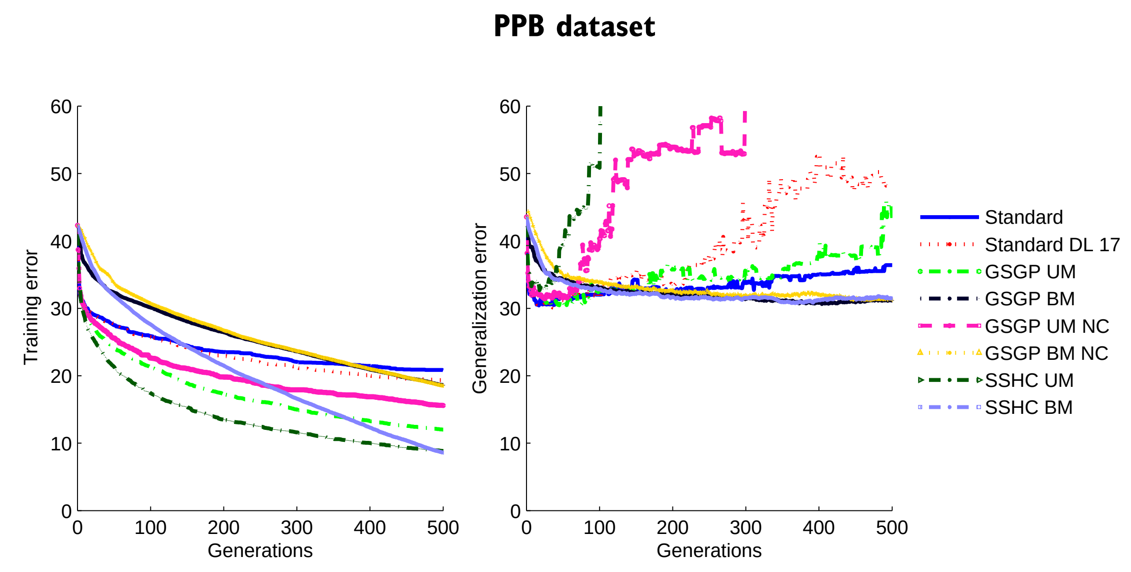
Vanneschi et al. (2013) mentioned that GSGP requires a relatively high mutation probability in order to explore the search space more efficiently. Indeed, the results of this section show only small differences between using the crossover operator (GSGP UM and BM) or not using it at all (GSGP UM NC and BM NC). Statistically, there are no differences in terms of generalization, in any comparison with the same mutation operator. In terms of training error the results are not consistent: GSGP BM NC is significantly better than GSGP BM in the Bio dataset (p-value 3.955×10^{-5}); GSGP UM is significantly better than GSGP UM NC in the PPB (p-value 4.734×10^{-11}) and LD50 (p-value 3.940×10^{-3}) datasets; and no other statistically significant differences were found. Figure 5.8 shows the evolution of the geometric semantic crossover effectiveness across the generations. The effectiveness of the standard crossover used in Standard GP is also shown. In this context, effectiveness is to be understood as the percentage of crossover applications that are able to generate an individual which is superior to another reference individual.

Particularly, the left column of figure 5.8 shows the percentage of crossover applications that are able to generate an individual which is superior to the current best individual in the population. To facilitate the description, this effectiveness will be referred to as the improvement over the best (IOB). The right column of figure 5.8 shows the percentage of crossover applications that are able to generate an individual which is superior to the best parent among the two parents selected for that particular crossover application. This effectiveness will be referred to as the improvement over the (best) parent (IOP). The data from figure 5.8 shows that the geometric semantic crossover is very ineffective in generating new overall best individuals. This is clear from the fact that the IOB is below 5% most of the time. This consideration is also applicable for the standard crossover. In terms of generating better individuals than the parents, the results are slightly better. The IOP can be around 30% in the best case. Notice that the Standard GP version without depth limit is slightly more effective in terms of IOP than the version with the depth limit. This might be related to the fact that the version with the depth limit must exclude the individuals generated with an invalid depth. This lowers the effectiveness of the operator. However, a possible ineffectiveness of the geometric semantic crossover should be expected. This operator can only produce an offspring which improves over both parents when the target semantics are between (even if partially) the semantics of the parents. Without an explicit semantic diversity control of the population and a mate selection procedure that takes the target semantics into account, the crossover operator may be ineffective. This ineffectiveness may also increase with larger semantic spaces, i.e., as the number of data instances increases. From these experiments, it can be concluded that the crossover operator can be skipped altogether since it does not significantly and consistently improve the search outcome (in generalization or training error). It also presents the disadvantage of exponentially increasing the size of the individuals, as opposed to the linear increase with the mutation operator.

On the other hand, the geometric semantic mutation is rather effective in generating better individuals. Figure 5.9 shows the evolution of the geometric semantic mutation effectiveness across the generations. The effectiveness of the standard mutation used in Standard GP is also shown. Similarly as with the crossover effectiveness measures, the left column of figure 5.9 shows the percentage of mutation applications that are able to generate an individual which is superior to the current best individual in the population. The right column of figure 5.9 shows the percentage of mutation applications that are able to generate an individual which is superior to the parent used for that particular mutation application. Starting with the mutation IOB, it is clear that when the geometric semantic mutation is applied to the current best individual (e.g., SSHC BM), the operator is very effective in generating new overall best individuals. If the mutation step is suitable for the current distance to the target, then this operator achieves an IOB of around 50% (as in the LD50 dataset). If throughout the run, the mutation step starts to become too large in regards to the distance to the target, then the IOB naturally starts to drop (as in the Bio and PPB datasets). This can be counteracted by adapting the mutation step throughout the run. This approach is explored in the next section. The BM also plays an effect here given that SSHC BM and SSHC UM achieve considerably different results. The IOB of SSHC BM is considerably superior to that of SSHC UM. Even when taking into account that SSHC UM fits the training data faster, it is still possible to notice a positive effect introduced by the bounding function. This is possibly achieved by smoothing the resulting semantics and by avoiding outliers for each mutation application. As with the standard crossover, the IOB of the standard mutation is also very low. Notice that for the SSHC variants, the IOB and the IOP are the same as the mutation is always applied to the current best individual. In terms of IOP, the BM variants present a similar behavior to the one described for SSHC BM in terms of IOB. Similarly, the UM variants also present a similar IOP behavior to the one described for SSHC UM in terms of IOB. The standard mutation presents a very inconsistent behavior in terms of IOP.

Finally, figures 5.10 and 5.11 show the tree size and depth evolution plots for the datasets tested. Starting with the tree size evolution, notice the exponential growth for the variants that use the geometric semantic crossover (GSGP UM and BM). The geometric semantic variants that only use the mutation operator present the expected linear growth. From these, the BM variants grow faster than the UM variants because the bounding functions added are naturally also included in the total number of nodes. Standard GP without depth limit presents an approximately linear growth. For this version of Standard GP, and in the Bio and PPB datasets, the median size of the individuals from the last generation is even higher than the median size of the individuals produced by GSGP UM NC and GSGP UM. In terms of tree depth, most methods present an approximately linear growth. It is interesting to notice that the Standard GP version without depth limit does not explore that much the possibility to reach very deep individuals. Instead, this version is more aggressive in terms of growing the number of nodes. However, and as





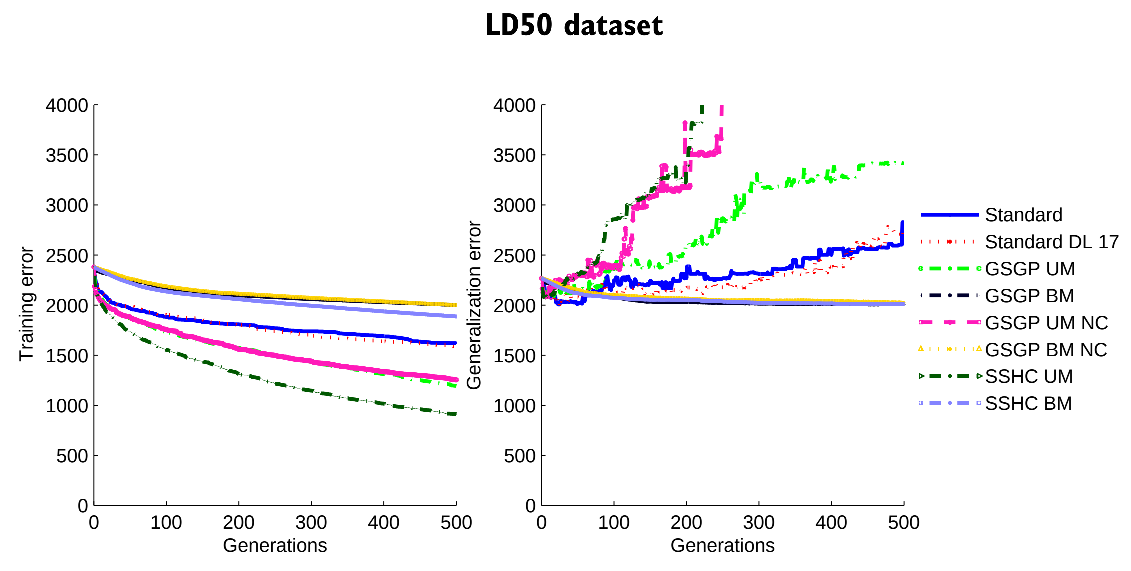


Figure 5.5: Training and generalization errors evolution plots for Standard GP, GSGP, and SSHC

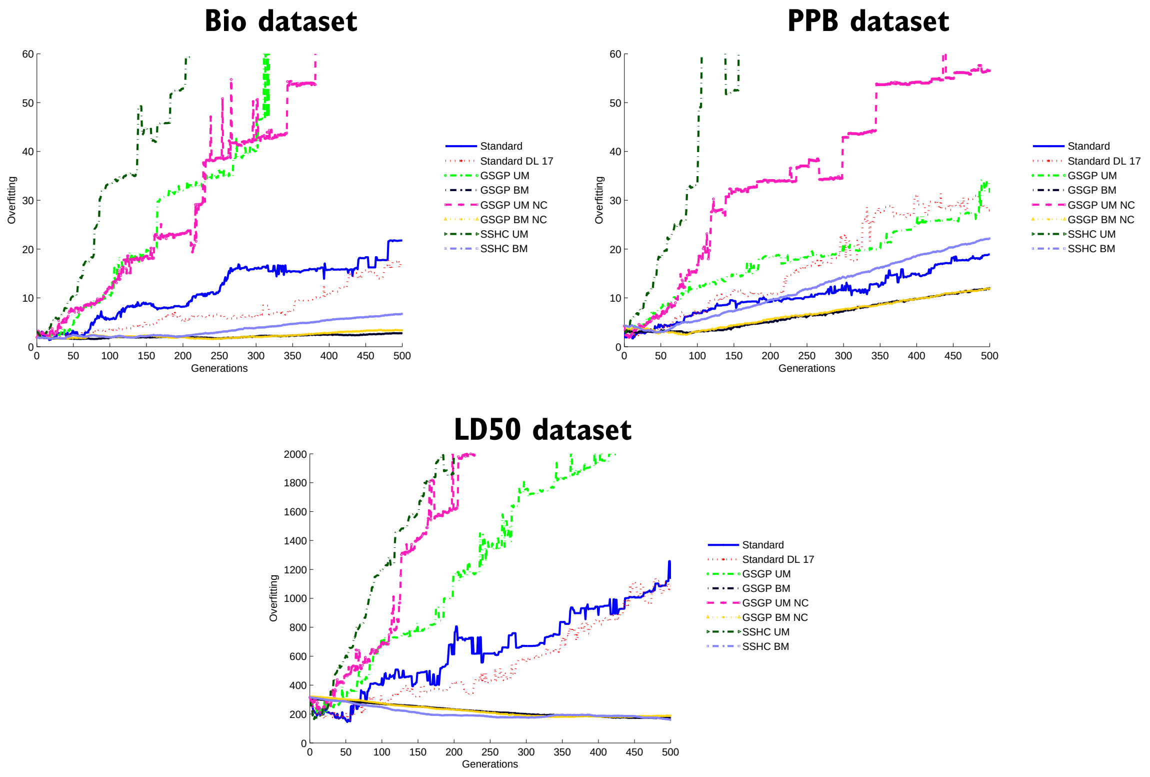


Figure 5.6: Overfitting evolution plots for Standard GP, GSGP, and SSHC

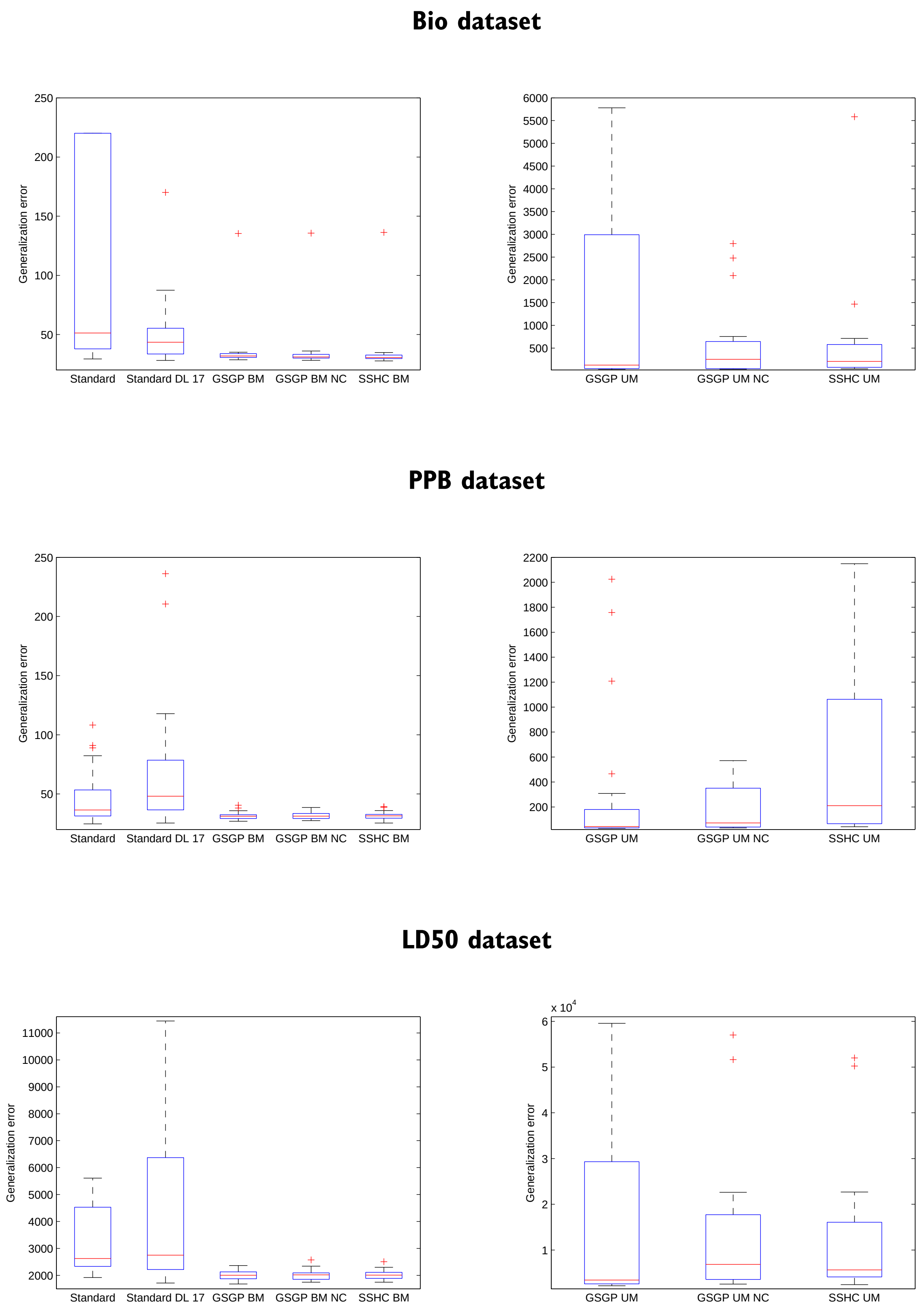


Figure 5.7: Generalization error boxplots for Standard GP, GSGP, and SSHC

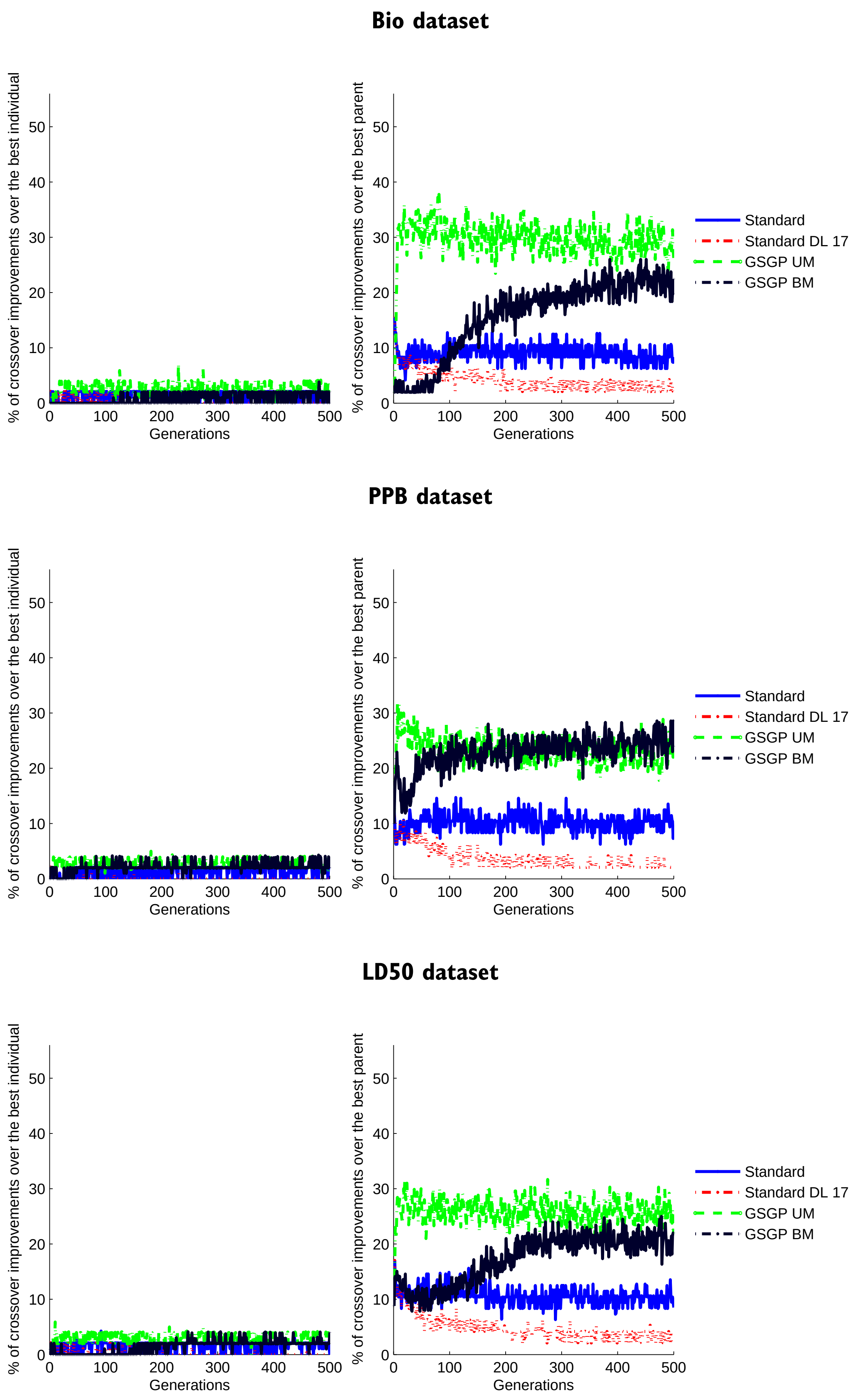


Figure 5.8: Percentage of crossover improvements for Standard GP and GSGP

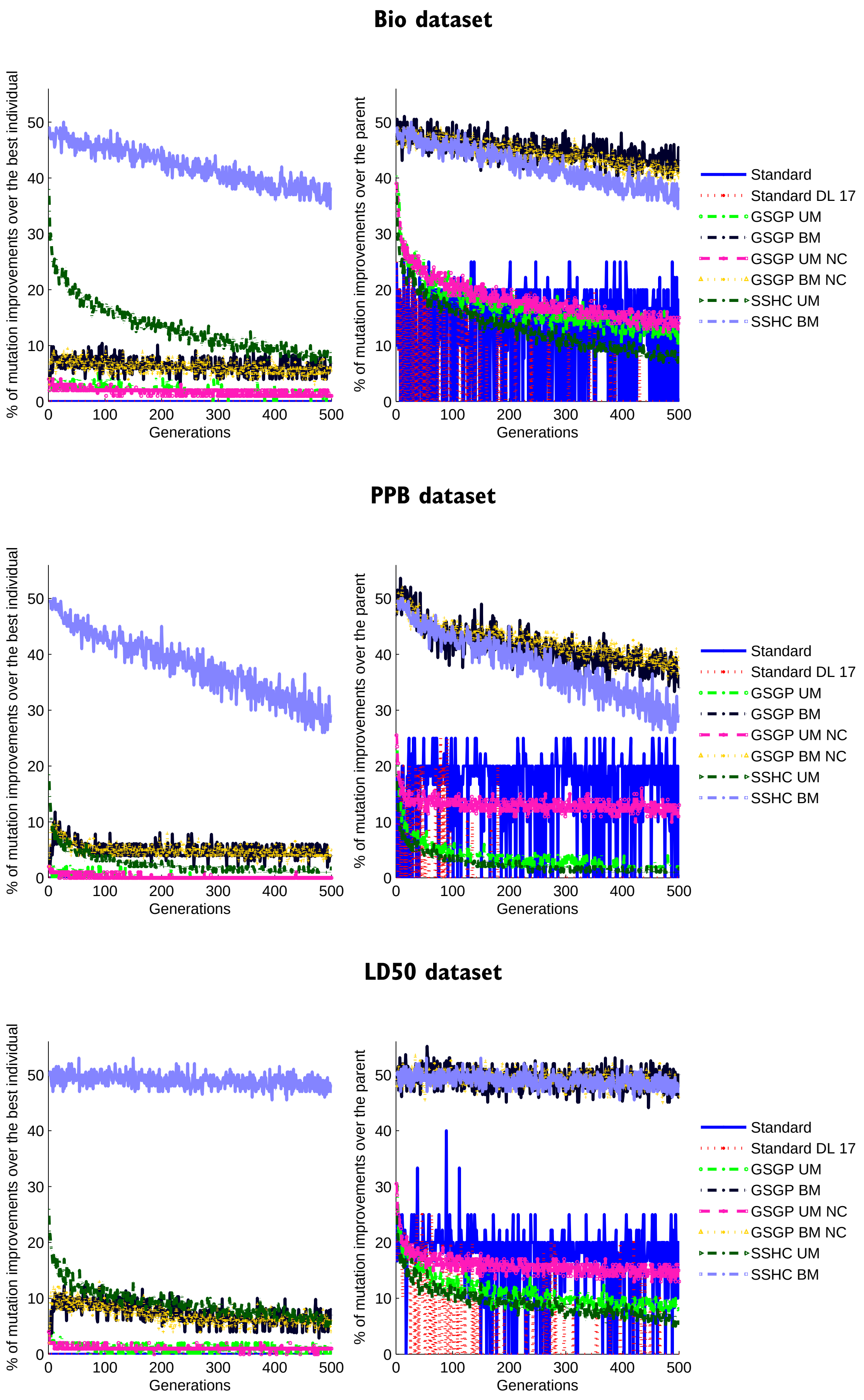


Figure 5.9: Percentage of mutation improvements for Standard GP, GSGP, and SSHC

mentioned previously, this freedom does not lead to better results when compared with the commonly imposed depth limit of 17.

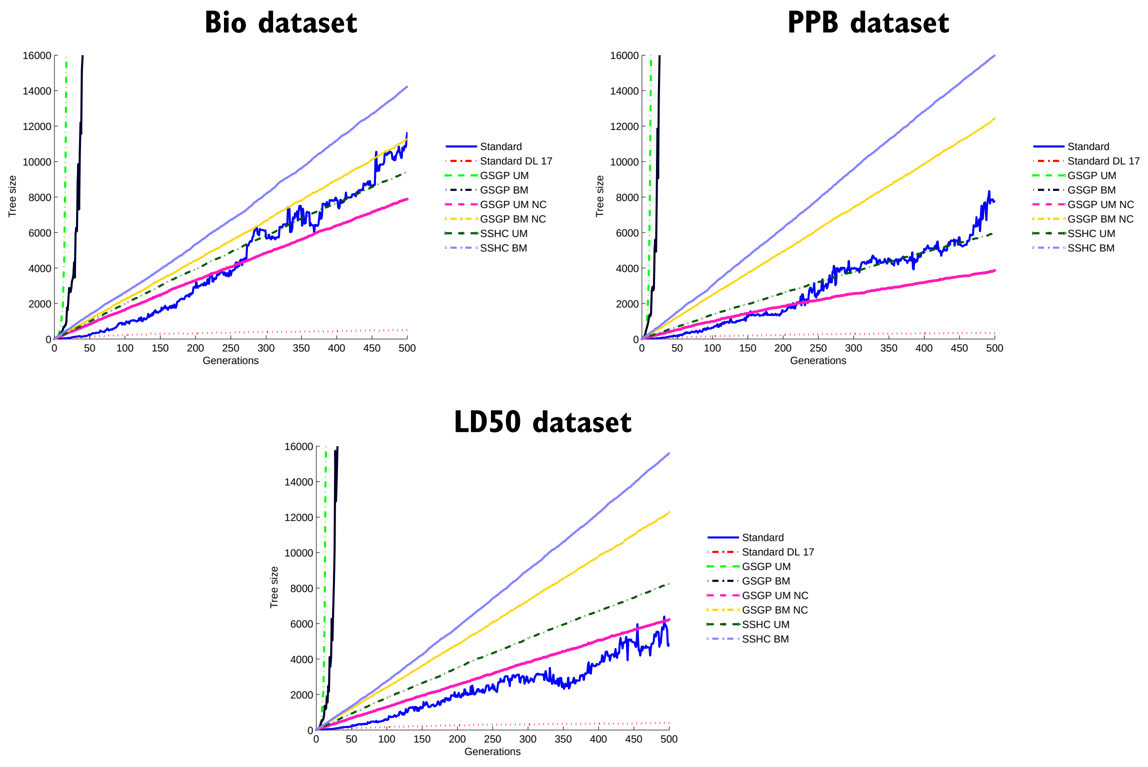


Figure 5.10: Tree size evolution plots for Standard GP, GSGP, and SSHC

5.2.3 Discussion

From the results of the previous subsections, it is clear that what differentiates the several geometric semantic variants in terms of generalization is the use of a bounded or unbounded mutation (BM and UM respectively). BM variants generalize well, while UM variants overfit the training data.

The GSGP implementation of Vanneschi et al. (2013) used a BM and reached the same conclusions regarding its competitive generalization. They justified this generalization ability by considering some properties of the geometric semantic operators. Particularly, they remarked that the geometric properties of these operators hold independently of the data in which the evaluation is taken place and, consequently, they also hold in unseen

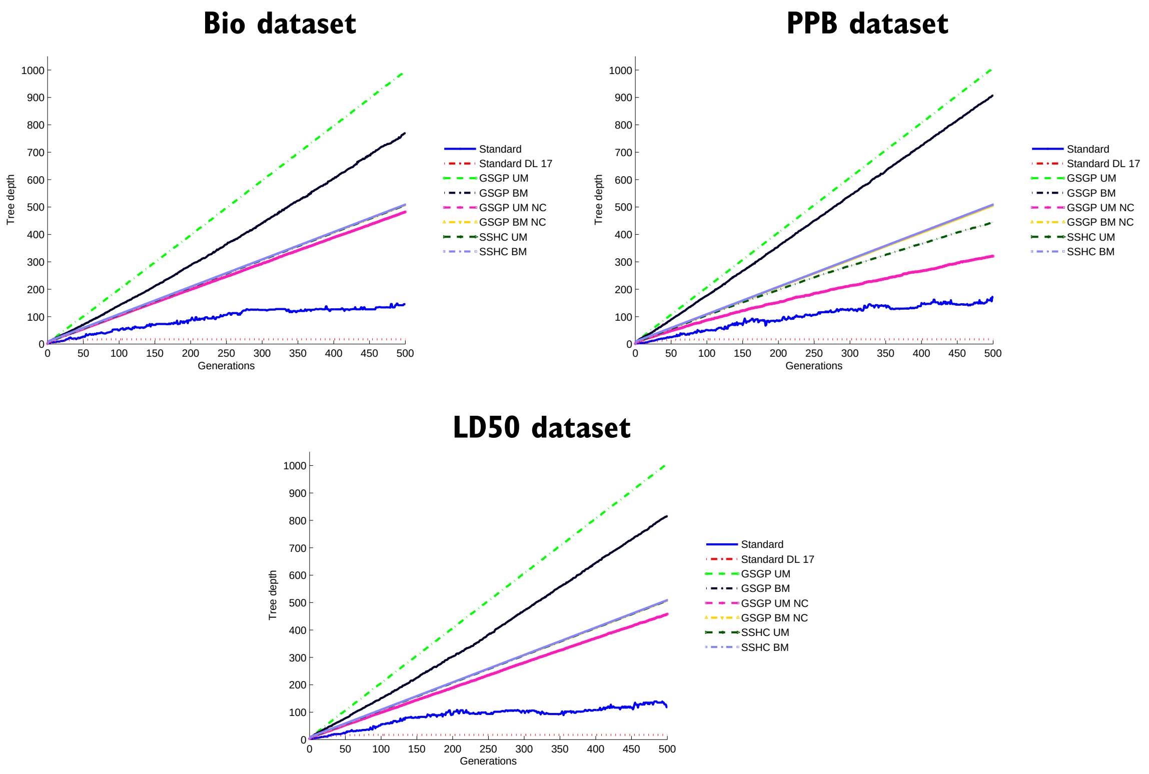


Figure 5.11: Tree depth evolution plots for Standard GP, GSGP, and SSHC

data. For the crossover operator this implies that each offspring produced also stands between its parents in the unseen data semantic space. Therefore, in the worst case, each offspring is not worse than the worst of its parents in unseen data. The implication for the mutation operator is that the perturbation that each offspring produces is bounded, also in the unseen data semantic space, by the mutation step (*ms*). Particularly, the semantic variation in the unseen data also ranges in the interval $[-ms, ms]$. Therefore, Vanneschi et al. (2013) concluded that the geometric semantic operators guarantee that a possible worsening of the generalization error is bounded and, consequently, that these operators help control overfitting.

As previously seen, the usage of a bounded or unbounded mutation was crucial in determining the generalization achieved. The BM operator was able to produce a competitive generalization by guaranteeing bounded and small perturbations in the unseen data. This was crucial to generalize well. However, it is clear that perturbations that increase the generalization error are always possible. It is also clear that if these perturbations were a significant majority of the applications of the operator then overfitting would be inevitable. Therefore, it can be concluded that after reaching what can be thought of as a generalization plateau (the point where it seems that no further induction can be performed with the available data), the BM operator generates about half of its perturbations in the decreasing generalization error direction and the other half in the increasing generalization error direction. These perturbations end up compensating each other, and therefore creating the relatively smooth generalization plateau. On the other hand, the UM operator performed badly in terms of generalization. Since in this operator the perturbations produced in the unseen data can be arbitrarily large, a single application of a mutation that results in overfitting (decreases training error but increases generalization error) can have an arbitrarily large increase in the generalization error. This results in considerable uncertainty in the generalization error evolution. This effect may be more noticeable in regression problems since any data instance can have an arbitrarily large error contribution, as opposed to classification problems where the error is commonly bounded for each data instance. For these reasons, the BM operator seems more robust and should be preferred over the UM operator.

For the reasons previously mentioned, the crossover operator had little effect in the results. However, in principle, the crossover operator should be riskier in terms of generalization than the BM operator. This is because the variation in the unseen data semantic space, although bounded by the unseen data semantics of the parents, can still be arbitrarily large. This results from the fact that the parents can be, in terms of unseen data semantics, very far apart. Since there is no way of knowing if the parents are close or far apart in the unseen data semantic space, the bounds (defined by the semantics of both parents) in unseen data are not useful in practice. This is another disadvantage of the crossover operator, following the exponential growth of the offspring produced and the low effectiveness in terms of search.

Although the generalization achieved by GSGP with bounded mutation is very competitive, the issue of the size of the solutions generated by these geometric semantic operators remains. As mentioned in the previous section, using crossover in GSGP translates into an exponential growth of the individuals. In the results of this section, individuals in GSGP reach thousands of nodes with less than 30 generations conducted. This raises the question: how can such large/complex individuals (models) achieve such competitive generalization? Some interpretations of theories such as Occam's razor and the Minimum Description Length principle state that smaller/less complex models generalize better. Consequently, and in light of these views, this result would be improbable, if not impossible. How can this be? A possible answer may lie in Ensemble Learning. Ensemble Learning is a Machine Learning paradigm in which several models are created and combined to produce a final model. Dietterich (2000) provided three reasons as to why constructing an ensemble of models may be superior to constructing a single model. The first two reasons are computational and representational. The computational reason is related to the difficulties in searching the search space, such as getting stuck in local optima. The representational reason arises when the true target function cannot be represented by any of the hypotheses in the search space. These first two reasons are not discussed in detail as they are not relevant to GSGP, respectively because the semantic space has no local optima and because in GSGP (and in traditional GP) any hypothesis can be represented that could also be represented by an ensemble (given suitable function and terminal sets). The last reason is the one which is relevant to GSGP and to generalization in general. It is a statistical reason and it is related to the fact that several different models can have a similar or even the same training data performance. This is essentially a model selection problem. Which model should be chosen? There is no way of knowing which model will generalize better. Ensemble Learning tackles this issue by combining several accurate models, which reduces the risk of the final model being overfitted. Even if some overfitted models are present in the ensemble, their negative contribution to the final model will be reduced since the final model will also include contributions from models which generalize well. It is a common result in Ensemble Learning to have large ensembles which achieve competitive generalization. Therefore, and in general, large/complex models (individuals) can also generalize well depending on how they are constructed.

GSGP can be seen as an Ensemble Learning method, since its operators always combine existing individuals independently to produce new individuals. The crossover operator combines two parents with a randomly generated individual, and the mutation operator combines one parent with another randomly generated individual (the individual which results from subtracting the two random subtrees). These parents and randomly generated individuals can be thought of as full models themselves. This interestingly relates back to Ensemble Learning, where a necessary condition for its positive outcome is that the ensemble must have a mix of accurate and diverse models (Hansen and Salamon, 1990). In GSGP, the parents can be seen as the accurate models (as they have survived during the evolution), and the randomly generated individuals as providing the also needed diversity. GSGP may derive some of its competitive generalization from this. For instance, let I_{2M} be the individual that results from two consecutive applications of the mutation operator to a given initial parent. It follows that:

$$I_{2M} = P + R1 \cdot ms + R2 \cdot ms, \tag{5.4}$$

where P is the initial parent, R1 and R2 are the two randomly generated individuals, and *ms* is the mutation step. Consequently, considering only the use of the mutation operator, GSGP can be seen as a weighted sum combination of models (it can be considered that the initial parent has a weight of 1).

In the end, GSGP successfully combines elements from Ensemble Learning (implicitly) and from the geometric semantic framework. Combining several models to incrementally produce new models has roots in Ensemble Learning. This allows to reduce the model selection risk by offsetting possible bad models. On the other hand, the combination of a structurally bounded mutation (BM) and a small mutation step can further reduce the issue of adding bad models by guaranteeing that their contribution will be small.

5.3 Optimal Weights and Their Effect on Generalization

As discussed in the previous section, the mutation step can play a role in reducing the risk of overfitting. On the other hand, when it comes to learning more efficiently, the geometric semantic mutation can be improved by adapting its step. It is possible to deterministically compute the optimal mutation step for each application of the operator. The geometric semantic mutation can be seen as a linear combination of two elements: the parent ***P***, and the random individual ***RI*** which results from subtracting the two random subtrees. Since ***RI*** is multiplied by the mutation step *ms*, a mutation step should be found such that:

$$P + RI \cdot ms = t, \tag{5.5}$$

where ***P*** and ***RI*** are semantic vectors and ***t*** is the target vector of the data. Since the parent is fixed and not influenced by any weight, the equation can be rewritten as:

$$RI \cdot ms = (t - P), \tag{5.6}$$

where a general linear system is reached:

$$A \cdot x = y, \tag{5.7}$$

the solution of which can be computed deterministically and optimally by the application of the Moore-Penrose pseudoinverse (hereafter simply inverse). This inverse computes the mutation step which minimizes the error in the training data for each specific combination of ***RI***, ***P*** and ***t***. This modification of the mutation operator is referred to as Adaptive Mutation (AM). As this work has studied the effects of bounded and unbounded mutations, the AM can be divided as: Adaptive Unbounded Mutation (AUM) and Adaptive Bounded Mutation (ABM). Moraglio and Mambrini (2013) had also remarked that the inverse could be used for this same purpose. They tested their approach in artificially generated problems. Here, this type of operator is used in real-world multidimensional datasets.

Following a similar reasoning, another mutation operator can be devised by adding a weight to the parent and adjusting both weights with the inverse. Let *pw* be the parent weight. Consequently:

$$P \cdot pw + RI \cdot ms = t \tag{5.8}$$

This new geometric semantic mutation operator will be called Doubly Adaptive Mutation (DAM), and it can also be divided as: Doubly Adaptive Unbounded Mutation (DAUM), and Doubly Adaptive Bounded Mutation (DABM). The inverse method could also be used to perform a linear combination of more than two weighted individuals.

These newly devised operators are tested with the SSHC (more efficient than GSGP as previously seen) and consequently its variants are named: SSHC AUM, SSHC DAUM, SSHC ABM and SSHC DABM. Figure 5.12 shows the training and generalization errors evolution plots for these adaptive variants. They reveal to be superior in terms of learning the training data when compared to the SSHC variants without adaptive mutation step (SSHC UM and SSHC BM). This was expected, since the mutation step adaptation is optimal for each application of the adaptive operators. In terms of generalization, these variants quickly overfit. In light of the analysis made previously, this quick overfitting should also be expected, since in these variants the weights can be arbitrarily large, and consequently, the benefits of using a structural bound (SSHC ABM and SSHC DABM) are lost. In other words, the overfitting risk reduction mechanism is lost without a small mutation step.

However, an interesting property can be found when looking closely at the initial generations. Figure 5.13 presents the generalization error evolution for the first 10 generations. It shows that these variants achieve a competitive generalization in only a single application of the mutation operators. This is particularly clear in the SSHC DAUM and SSHC DABM variants. These two variants were expected to fit the training data more easily when compared to the other two variants (SSHC AUM and SSHC ABM), since they have an extra degree of freedom (the parent weight). Further testing is needed to determine if this property holds across other datasets. If it holds, then these mutation variants become a competitive alternative when performing semantic search, particularly since they produce small individuals and compute fast. They also raise no issues in constructing/reconstructing large individuals, as opposed to what may happen with the GSGP variants.

Similarly as in the previous section, figure 5.14 shows the effectiveness of the adaptive mutation operators across the generations. All variants are extremely effective in

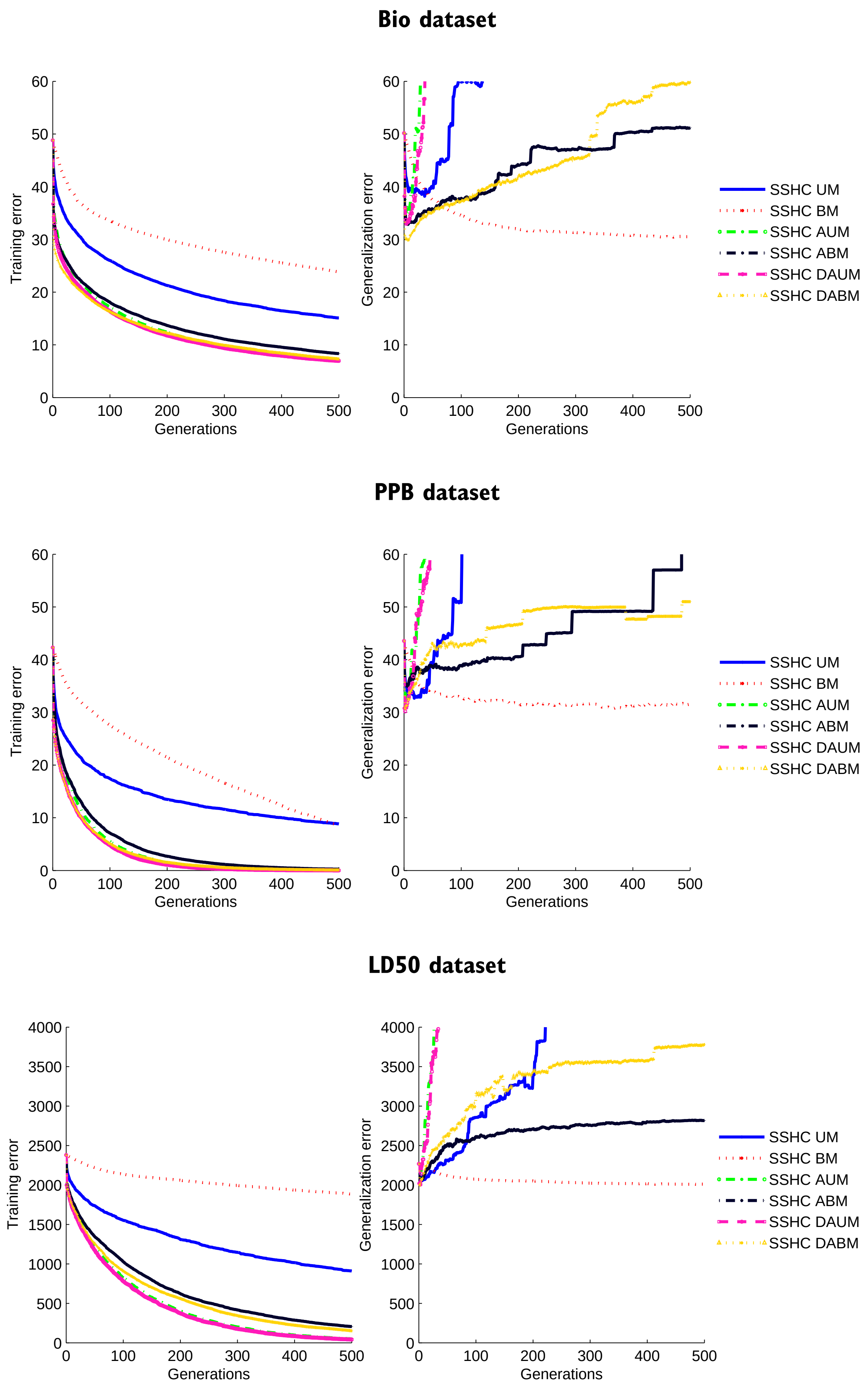


Figure 5.12: Training and generalization errors evolution plots for SSHC with and without adaptive mutation step

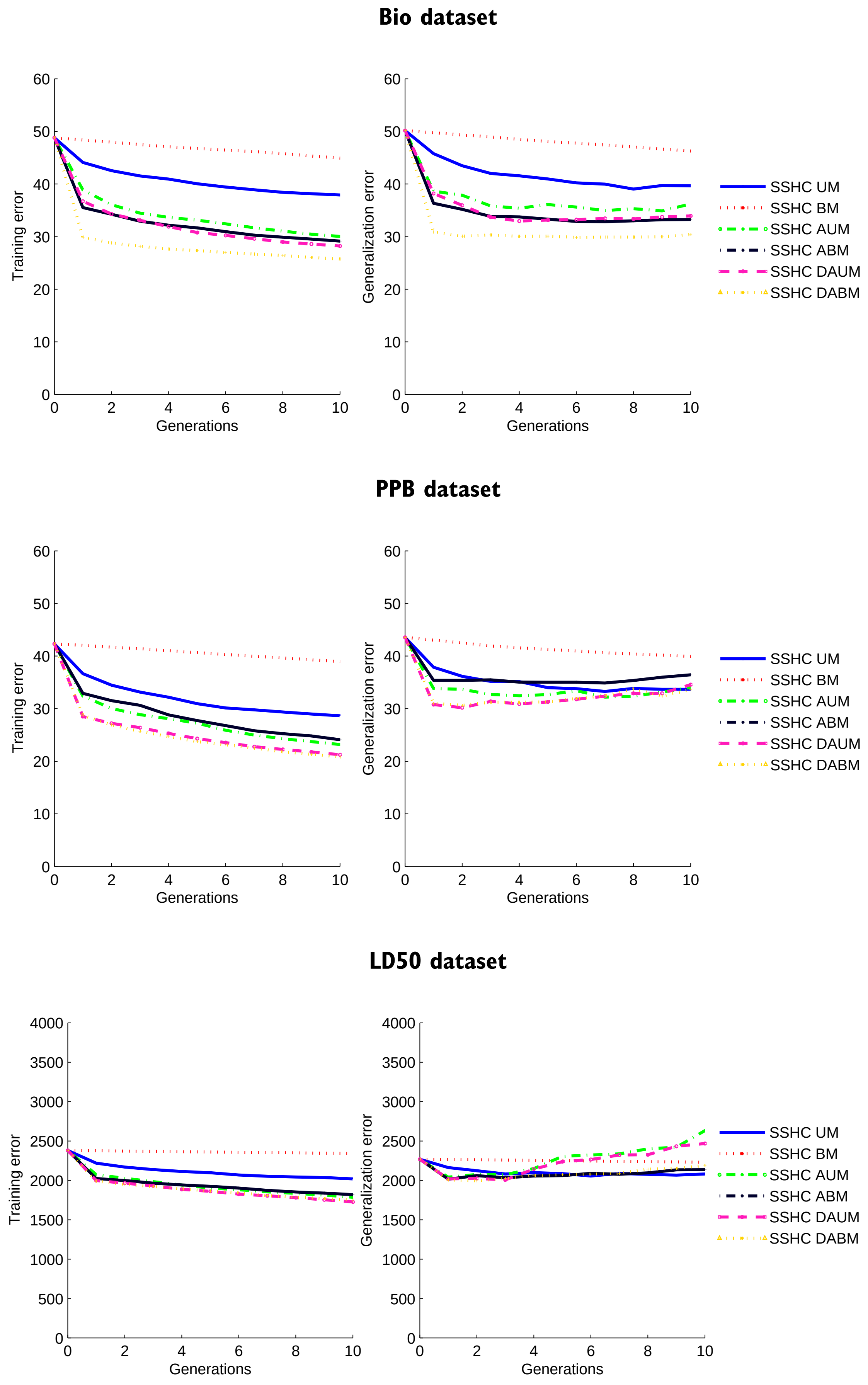


Figure 5.13: Training and generalization errors evolution plots for the first 10 generations for SSHC with and without adaptive mutation step

improving over the current best individual. The percentage of successful mutations is at least 93% for all datasets. In the LD50 dataset, it was even possible to always generate a better individual. The behavior is almost the same in the PPB dataset, with just some minor exceptions with 99% success as opposed to 100%. Since the Bio dataset is slightly noisy, the values drop to 93% in the worst case, but the most common values are 95% and 96%.

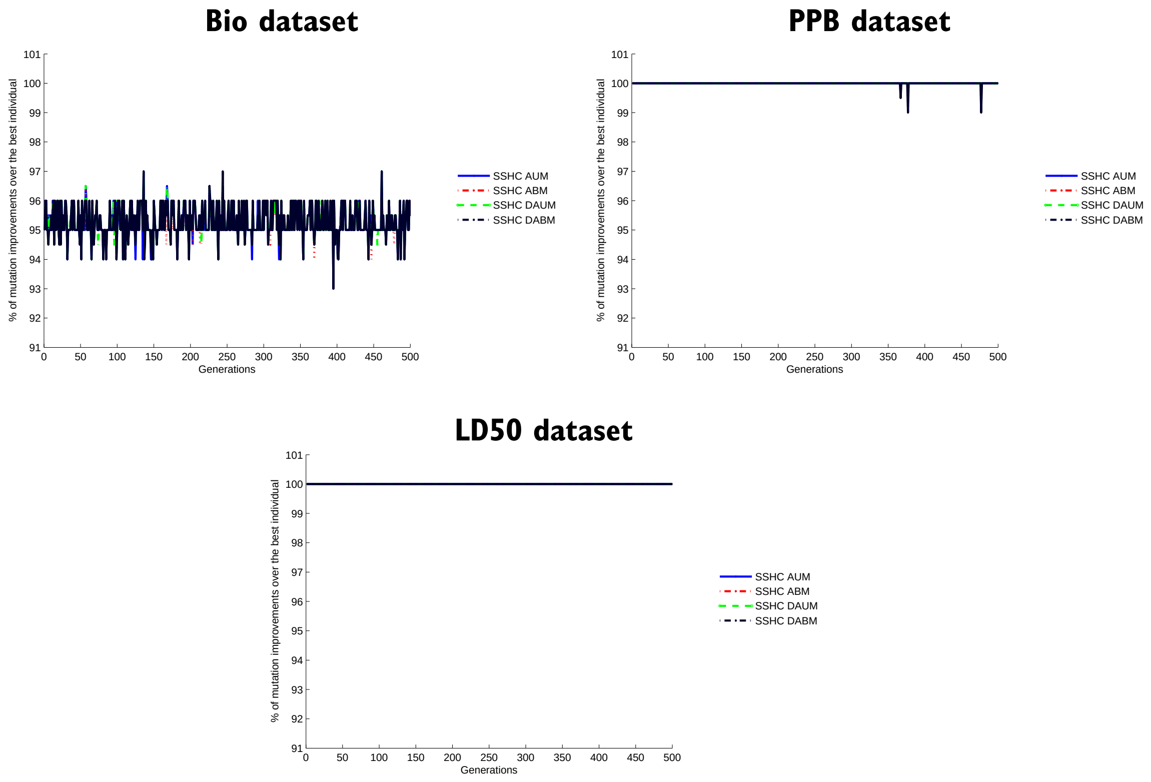


Figure 5.14: Percentage of mutation improvements for SSHC with adaptive mutation step

5.4 Arbitrarily Close Alignments in the Error Space

In line with the work being conducted in semantics in GP, a recent work by Ruberto et al. (2014) explored the possibility of reaching optimal individuals in terms of training data by searching for aligned or coplanar individuals in the error space. As it will be made clear ahead, this notion of error space is closely related to the notion of semantic space.

5.4.1 Error Space Regularities

Following and adapting from Ruberto et al. (2014), let $\mathbf{X} = \{\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_n}\}$ be the set of input data of a supervised learning problem, and $\overrightarrow{t} = [t_1, t_2, ..., t_n]$ the vector of the respective target values, i.e., t_i is the expected value for input instance $\overrightarrow{x_i}$. In the supervised learning context, a GP individual (or a model in general) is a function that computes an output for each input instance $\overrightarrow{x_i}$. As mentioned before, these outputs define the semantics of an individual. Let $I(\overrightarrow{x_i})$ be the output of an individual I to a given input instance $\overrightarrow{x_i}$. Consequently, the semantics of an individual can be defined as the vector $\overrightarrow{s_I} = [I(\overrightarrow{x_1}), I(\overrightarrow{x_2}), ..., I(\overrightarrow{x_n})]$. This vector represents a point in the n-dimensional semantic space. The size of the semantic space (n) is defined by the number of input instances, i.e., the size of the set $\mathbf{X}$. The error vector of an individual can be constructed from the semantics vector by taking into account the target vector:

$$\overrightarrow{e_I} = \overrightarrow{s_I} - \overrightarrow{t}. \tag{5.9}$$

This vector represents a point in the n-dimensional error space. From the definition of the error vector, it follows that the semantic optimum translates into the origin of the error space. Notice that the mapping between the elements of the semantic space and the elements of the error space is bijective, i.e., for each possible point in the semantic space there is exactly one corresponding point in the error space, and vice versa.

Particular regularities in the error space can be explored to analytically construct an optimal individual in terms of training data. Ruberto et al. (2014) showed that this can be achieved by finding two aligned individuals or three coplanar individuals. They named their approaches to finding such regularities, respectively, as ESAGP-1 and ESAGP-2. In these approaches the search is modified to find these regularities as opposed to directly minimizing the training error. Ruberto et al. (2014) tested their approaches in two real-world multidimensional datasets. Although both versions of ESAGP fit the training data faster than Standard GP, they were still very far from reaching aligned or coplanar individuals. Consequently, the question of how to effectively find these regularities in the error space remained unanswered. An important related question is that of generalization to unseen data. It is still unclear how would the individuals created from the aligned or coplanar individuals behave in terms of generalization. Would these individuals generalize well or would they overfit?

In this section a different approach to finding aligned individuals is proposed. Following and adapting from Ruberto et al. (2014), the definition of optimally aligned individuals is the following:

Definition 3 Optimally Aligned Individuals: Two GP individuals A and B are optimally aligned if there is a scalar constant k such that: $\overrightarrow{e_A} = k \cdot \overrightarrow{e_B}$,

where k is an error vector proportionality constant. The alignment of two individuals (A and B) can be computed by the absolute cosine similarity between their respective error vectors:

$$\text{Absolute cosine similarity} = |\cos\theta| = \left|\frac{\overrightarrow{e_A} \times \overrightarrow{e_B}}{||\overrightarrow{e_A}|| \cdot ||\overrightarrow{e_B}||}\right| \tag{5.10}$$

where $\times$ represents the scalar product between two vectors, and $||.||$ represents the Euclidean norm of a vector. Two optimally aligned individuals will have an absolute cosine similarity of exactly 1, and two orthogonal individuals (i.e., with an angle of 90°) will have an absolute cosine similarity of exactly 0. Figure 5.15 shows an example of 3 aligned individuals in the error space. Notice that the individual represented by C is aligned with A and B, even though it is in the other side of the error space, i.e., with an angle of 180°. This translates into a negative k. Since in terms of alignment it is irrelevant if both individuals are in the same side or not, the latter case should not be penalized. This is why the absolute value is considered for the cosine similarity.

If two aligned individuals (A and B) are found, then an optimal individual in terms of training data can be analytically constructed. By applying the error vector definition from equation 5.9, the aligned individuals equation can be rewritten as:

$$\overrightarrow{s_A} - \overrightarrow{t} = k \cdot (\overrightarrow{s_B} - \overrightarrow{t}). \tag{5.11}$$

It follows that:

$$\overrightarrow{t} = \frac{1}{1-k} \cdot \overrightarrow{s_A} - \frac{k}{1-k} \cdot \overrightarrow{s_B}, \tag{5.12}$$

from which an optimal individual in terms of training data can be constructed by creating the individual:

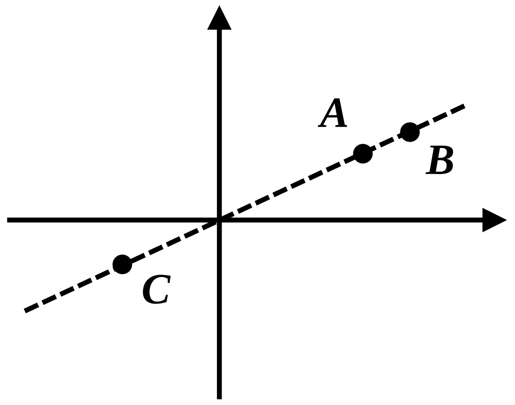


Figure 5.15: An example of 3 aligned individuals in the error space

$$I_{opt} = \frac{1}{1-k} \cdot A - \frac{k}{1-k} \cdot B \tag{5.13}$$

This analytic construction can be thought of as a type of semantic crossover. This crossover is not geometric as, depending on the value of *k*, the resulting individual may not be between the parents in the semantic space. Even if two individuals are not optimally aligned, this semantic crossover can still be used by using an approximation of *k*. This can be done by, for instance, using the median of the values $\overrightarrow{e_A}_1/\overrightarrow{e_B}_1$, $\overrightarrow{e_A}_2/\overrightarrow{e_B}_2, \ldots, \overrightarrow{e_A}_n/\overrightarrow{e_B}_n$. This was the approach used by Ruberto et al. (2014) and it is also the approach used here.

5.4.2 Arbitrarily Close Alignments

The approach proposed here is based on the fact that a target semantics that guarantees an optimal alignment can be easily defined. Consequently, the search can be modified to find individuals that produce this new target semantics, as opposed to searching directly for individuals that fit the original targets. By choosing a given individual and a given error vector proportionality constant (*k*), it is immediately defined what is the semantics of the other individual needed to reach an optimal alignment. Concretely, given any individual *A* and any error vector proportionality constant *k*, any individual *B* is optimally aligned with

A if defined by the following error vector:

$$\overrightarrow{e_B} = \frac{\overrightarrow{e_A}}{k}, \tag{5.14}$$

and, consequently, by the following semantic vector:

$$\overrightarrow{s_B} = \overrightarrow{e_B} + \overrightarrow{t}. \tag{5.15}$$

Notice that A together with the origin of the error space define the line, and k, defines the exact target point in the same line. Referring back to figure 5.15, if A was chosen as the initial point, then a particular positive k would lead to having B as the target, and a particular negative k would lead to C as the target. Knowing now the definition of $\overrightarrow{s_B}$, the search can be redefined to find an individual that produces this semantics. By achieving this, an optimal alignment is found and, consequently, an optimal individual in terms of training data can be constructed. It is important to emphasize that this redefinition of the target semantics does not change the nature of the semantic space. In particular, by using the geometric semantic operators, the search process can still take advantage of the unimodality of the semantic space. Since this space has no local optima, and given that the geometric semantic mutation is an effective operator (as shown in the previous sections), the search can be focused around the current best individual. In other words, a geometric semantic hill climber is an effective and efficient strategy for conducting the search. For these reasons, the approach proposed here uses a SSHC to explore the search space. This approach will be referred to as ACA-SSHC: Arbitrarily Close Alignments (ACA) - Semantic Stochastic Hill Climber (SSHC). The general steps of the proposed approach are the following:

1. Set the error vector proportionality constant k
2. Choose an individual (A) from the initial random population
3. With k and A compute the target semantics ($\overrightarrow{s_B}$) using equations 5.14 and 5.15
4. From the initial random population choose the individual (T) which is closer to $\overrightarrow{s_B}$
5. Repeat the following operations until a given stopping criterion is met:

   5.1. Having $\overrightarrow{s_B}$ as the target, apply the geometric semantic mutation to T N times to generate N new individuals

   5.2. Update T as being the closest individual to $\overrightarrow{s_B}$, selected from the current T and the N newly generated individuals

6. Apply the semantic crossover from equation 5.13 to A and T to create the final individual (B')

The choice of A is not particularly influential as it simply helps to define the alignment line. In ACA-SSHC, A is chosen as the best individual in terms of training error from the initial population. It could also be chosen randomly or by using some other criterion. Notice that A does not change during the search. The mentioned closeness to the new target semantics ($\overrightarrow{s_B}$) is computed by a given error measure as it would be for the original targets. In ACA-SSHC the root mean squared error is used to measure the closeness to the new target semantics. The stopping criterion used in ACA-SSHC is based on the training error of the individual that can be constructed analytically from A and T. When this training error falls below a given threshold the search is stopped. The stopping criterion could also be based on the absolute cosine similarity between A and T. Notice that the final individual (B') is an approximation of the target semantics ($\overrightarrow{s_B}$). In theory, this approximation can be as arbitrarily close as desired. In practice, given the real-valued nature of the semantics considered here, an optimal alignment may in fact require exhaustive computation, and may potentially lead to issues in the floating-point arithmetic.

5.4.3 Results and Discussion

For the experimental testing, ACA-SSHC is compared against a SSHC that searches directly for the original targets. Both approaches use DABM as the mutation operator. Also for both approaches, the stopping criterion is based on a training error threshold. For the PPB and LD50 datasets, the threshold used is 1.0, and for the Bio dataset the threshold used is 5.0. The experiments in the Bio dataset are computationally more expensive because this dataset is noisier than the other two. For this reason, the training error threshold used is sightly higher. However, it was confirmed in preliminary tests that using 1.0 or lower values is still feasible. Notice that since the stopping criterion is based on the training error, the number of generations is variable. The values of k tested are: 0.1, 0.25, 0.5, 0.75, 0.9, 1.1, 1.25, 1.5, 1.75, 2.0, and their corresponding negative values. All the other parameters are the same as the ones used in the previous sections.

Starting with ACA-SSHC, the results that show that this approach is effective in finding near-optimal alignments in the error space are first presented. Tables 5.2 to 5.4 show the median, average, and standard deviation (SD) of the absolute cosine similarity between A and T for each k and for each dataset. As it can be seen, near-optimal alignments can be achieved in all datasets and for all values of k. The variance between the results of different values of k is negligible. It seems that the search performance is unaffected by the parameter k. This behavior is consistent across the datasets tested.

Figure 5.16 shows the evolution of the alignment throughout the generations when using $k = 1.1$. This value of k was chosen arbitrarily since all values of k have similar behavior. The values presented in the evolution plots are medians over the 30 runs. From these plots it is possible to see that the evolution of the alignment is rather smooth. This shows that ACA-SSHC is effective and efficient in finding the desired alignments. Even closer alignments could be considered by having a stricter stopping criterion. From the results presented here and given the nature of the semantic space, it is safe to assume that these even closer alignments would not present an issue in terms of search.

Tables 5.5 to 5.7 show the generalization error and the number of generations taken to achieve the desired alignment in all datasets. These results show that the individuals constructed from near-optimal alignments are prone to overfitting, as the values of the generalization errors are rather high. In comparable experiments and as seen in the previous sections, the best median generalization error achieved for these datasets was around 30 RMSE for PPB and Bio, and around 2000 for LD50. As shown in the results for the absolute cosine similarity, it seems that k has no particular influence in the search outcome. Although there is some variation for a few values, there is no general claim that can be made on how the value of k influences the results. This is also true when comparing the positive and negative values of the same k. It seems that the results are relatively independent of the choice of k, both for generalization error and number of generations taken.

Table 5.8 shows the generalization error and the number of generations taken to achieve the desired training error threshold for SSHC. SSHC presents similar results to ACA-SSHC in the sense that it also overfits the training data. This shows that the issue

Table 5.2: Absolute cosine similarity achieved in the Bio dataset

k	**Absolute cosine similarity**		
	Median	**Average**	**SD**
0.10	0.99363	0.99435	0.00130
-0.10	0.99575	0.99622	0.00087
0.25	0.99178	0.99270	0.00168
-0.25	0.99706	0.99738	0.00060
0.50	0.98817	0.98949	0.00242
-0.50	0.99869	0.99884	0.00027
0.75	0.98398	0.98573	0.00328
-0.75	0.99968	0.99971	0.00007
0.90	0.98112	0.98322	0.00385
-0.90	0.99995	0.99995	0.00001
1.10	0.97693	0.97953	0.00470
-1.10	0.99995	0.99995	0.00001
1.25	0.97361	0.97654	0.00536
-1.25	0.99968	0.99972	0.00006
1.50	0.96763	0.97116	0.00655
-1.50	0.99873	0.99887	0.00025
1.75	0.96096	0.96529	0.00786
-1.75	0.99715	0.99747	0.00057
2.00	0.95375	0.95896	0.00923
-2.00	0.99496	0.99552	0.00099

Table 5.3: Absolute cosine similarity achieved in the PPB dataset

k	**Absolute cosine similarity**		
	Median	**Average**	**SD**
0.10	0.99967	0.99964	0.00011
-0.10	0.99978	0.99976	0.00007
0.25	0.99957	0.99953	0.00014
-0.25	0.99984	0.99983	0.00005
0.50	0.99940	0.99933	0.00020
-0.50	0.99993	0.99993	0.00002
0.75	0.99916	0.99909	0.00028
-0.75	0.99998	0.99998	0.00001
0.90	0.99902	0.99893	0.00033
-0.90	9.999973e-01	9.999970e-01	9.058755e-07
1.10	0.99879	0.99868	0.00040
-1.10	9.999973e-01	9.999970e-01	8.983940e-07
1.25	0.99861	0.99848	0.00047
-1.25	0.99998	0.99998	0.00001
1.50	0.99831	0.99814	0.00056
-1.50	0.99993	0.99993	0.00002
1.75	0.99793	0.99774	0.00068
-1.75	0.99985	0.99983	0.00005
2.00	0.99758	0.99734	0.00080
-2.00	0.99973	0.99970	0.00009

Table 5.4: Absolute cosine similarity achieved in the LD50 dataset

k	Absolute cosine similarity		
	Median	**Average**	**SD**
0.10	9.999999e-01	9.999999e-01	1.613930e-08
-0.10	9.999999e-01	9.999999e-01	1.020797e-08
0.25	9.999999e-01	9.999999e-01	1.916355e-08
-0.25	1.000000e+00	9.999999e-01	6.968919e-09
0.50	9.999998e-01	9.999998e-01	2.482887e-08
-0.50	1.000000e+00	1.000000e+00	3.515140e-09
0.75	9.999997e-01	9.999997e-01	4.540934e-08
-0.75	1.000000e+00	1.000000e+00	7.415154e-10
0.90	9.999997e-01	9.999997e-01	4.811919e-08
-0.90	1.000000e+00	1.000000e+00	1.261224e-10
1.10	9.999996e-01	9.999996e-01	5.527597e-08
-1.10	1.000000e+00	1.000000e+00	1.179168e-10
1.25	9.999996e-01	9.999995e-01	6.680947e-08
-1.25	1.000000e+00	1.000000e+00	8.811649e-10
1.50	9.999995e-01	9.999994e-01	7.520555e-08
-1.50	1.000000e+00	1.000000e+00	2.868483e-09
1.75	9.999993e-01	9.999993e-01	8.684608e-08
-1.75	1.000000e+00	9.999999e-01	7.895287e-09
2.00	9.999992e-01	9.999992e-01	1.080166e-07
-2.00	9.999999e-01	9.999999e-01	1.141425e-08

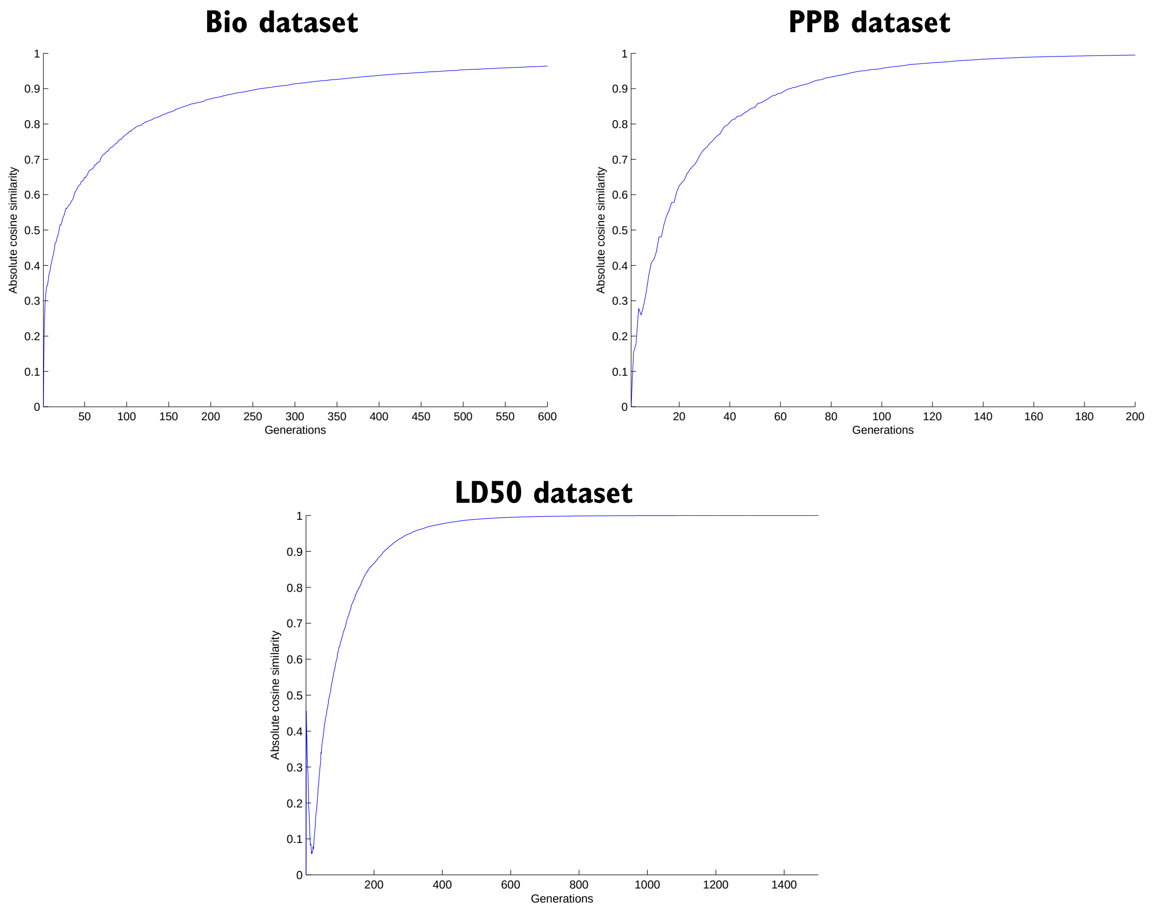


Figure 5.16: Absolute cosine similarity evolution for ACA-SSHC with $k = 1.1$

Table 5.5: Generalization error and number of generations taken to achieve the desired alignment in the Bio dataset

k	**Generalization error**			**Generations**		
	Median	**Average**	**SD**	**Median**	**Average**	**SD**
0.10	254.9	4.8e+08	2.5e+09	1151.5	1206.0	356.9
-0.10	220.3	4.2e+06	1.8e+07	1200.5	1225.4	349.6
0.25	112.3	4.2e+11	2.3e+12	1238.5	1211.5	364.9
-0.25	312.4	1.8e+08	8.9e+08	1242.5	1270.2	388.2
0.50	216.6	2.8e+05	1.2e+06	1230.0	1184.6	364.9
-0.50	205.1	3.0e+11	1.6e+12	1257.0	1322.1	399.9
0.75	220.7	5.1e+06	2.4e+07	1190.0	1171.3	340.1
-0.75	354.7	4.6e+30	2.5e+31	1361.5	1314.8	369.3
0.90	118.4	9.6e+06	5.2e+07	1244.5	1185.7	352.4
-0.90	593.1	3.5e+07	1.2e+08	1491.0	1433.4	391.5
1.10	294.9	4.4e+05	2.4e+06	1092.0	1157.3	338.1
-1.10	185.6	4.0e+08	2.1e+09	1330.5	1318.7	313.1
1.25	93.9	5.5e+05	2.8e+06	1123.5	1158.2	338.1
-1.25	317.9	3.9e+11	2.1e+12	1183.5	1152.1	295.0
1.50	138.8	3.3e+08	1.6e+09	1200.5	1178.6	363.6
-1.50	112.1	2.2e+06	9.0e+06	1048.0	1084.6	305.1
1.75	160.6	2.8e+08	1.5e+09	1126.0	1150.3	326.5
-1.75	182.6	1.8e+07	9.7e+07	1066.0	1081.4	297.2
2.00	113.9	1.1e+11	6.0e+11	1231.5	1163.9	326.6
-2.00	211.8	3.7e+08	1.9e+09	1078.5	1053.8	280.8

Table 5.6: Generalization error and number of generations taken to achieve the desired alignment in the PPB dataset

k	Generalization error			Generations		
	Median	Average	SD	Median	Average	SD
0.10	45.2	4.8e+09	2.3e+10	287.0	291.7	34.4
-0.10	258.2	1.1e+08	5.6e+08	305.0	297.6	30.9
0.25	44.0	3.5e+09	1.9e+10	278.5	283.9	35.0
-0.25	49.9	2.1e+14	1.2e+15	289.0	295.0	25.3
0.50	41.8	4.4e+15	2.4e+16	276.5	274.6	22.2
-0.50	51.8	5.7e+11	3.1e+12	278.0	284.6	28.6
0.75	47.7	2.3e+44	1.3e+45	271.5	366.9	387.2
-0.75	72.6	1.8e+13	1.0e+14	273.5	273.5	35.0
0.90	55.2	6.6e+15	3.6e+16	266.0	342.9	315.5
-0.90	125.9	4.9e+11	2.7e+12	289.0	304.3	56.9
1.10	42.6	3.2e+07	1.5e+08	263.0	267.5	24.7
-1.10	100.3	6.8e+12	3.7e+13	286.5	301.1	57.3
1.25	40.7	3.6e+07	1.6e+08	258.5	262.2	28.9
-1.25	48.5	1.6e+13	9.0e+13	257.0	255.0	35.5
1.50	47.7	4.3e+33	2.3e+34	256.0	262.1	33.2
-1.50	47.1	1.8e+27	1.0e+28	240.0	241.5	26.0
1.75	64.7	8.0e+30	4.4e+31	262.0	263.6	24.4
-1.75	41.0	2.6e+27	1.4e+28	236.0	236.9	24.3
2.00	88.6	1.7e+09	9.1e+09	254.5	260.1	28.9
-2.00	44.6	2.4e+27	1.3e+28	235.5	241.6	23.3

Table 5.7: Generalization error and number of generations taken to achieve the desired alignment in the LD50 dataset

k	Generalization error			Generations		
	Median	**Average**	**SD**	**Median**	**Average**	**SD**
0.10	5156.9	1.7e+12	6.7e+12	2516.0	2623.3	533.0
-0.10	5969.6	4.2e+14	1.6e+15	2464.5	2593.8	581.6
0.25	9433.4	3.3e+56	1.8e+57	2529.5	2624.6	528.5
-0.25	7858.6	3.3e+56	1.8e+57	2565.0	2639.3	560.1
0.50	4418.0	6.1e+20	3.4e+21	2487.5	2576.8	573.2
-0.50	6036.8	3.1e+26	1.7e+27	2493.0	2670.3	712.3
0.75	9296.8	2.1e+11	7.2e+11	2553.5	2621.9	584.9
-0.75	18369.3	4.9e+26	2.7e+27	2418.5	2670.9	654.4
0.90	4591.4	3.0e+12	1.3e+13	2522.5	2548.5	533.4
-0.90	6374.3	4.5e+44	2.5e+45	2274.5	2531.0	683.8
1.10	5316.1	6.9e+49	3.8e+50	2408.5	2559.6	529.9
-1.10	6382.8	2.5e+27	1.4e+28	2192.0	2308.1	571.9
1.25	5157.5	1.8e+14	9.3e+14	2507.5	2601.5	527.9
-1.25	4342.3	9.2e+13	5.0e+14	2183.5	2211.0	440.6
1.50	8647.1	1.4e+15	7.8e+15	2392.0	2510.1	517.1
-1.50	5993.4	3.2e+30	1.8e+31	2335.5	2364.3	505.5
1.75	5022.2	7.8e+13	4.2e+14	2478.0	2569.7	522.5
-1.75	4226.0	3.9e+14	2.1e+15	2306.0	2347.9	506.8
2.00	4336.1	3.6e+62	2.0e+63	2426.0	2586.6	545.3
-2.00	5183.0	1.4e+14	7.6e+14	2272.0	2354.6	485.9

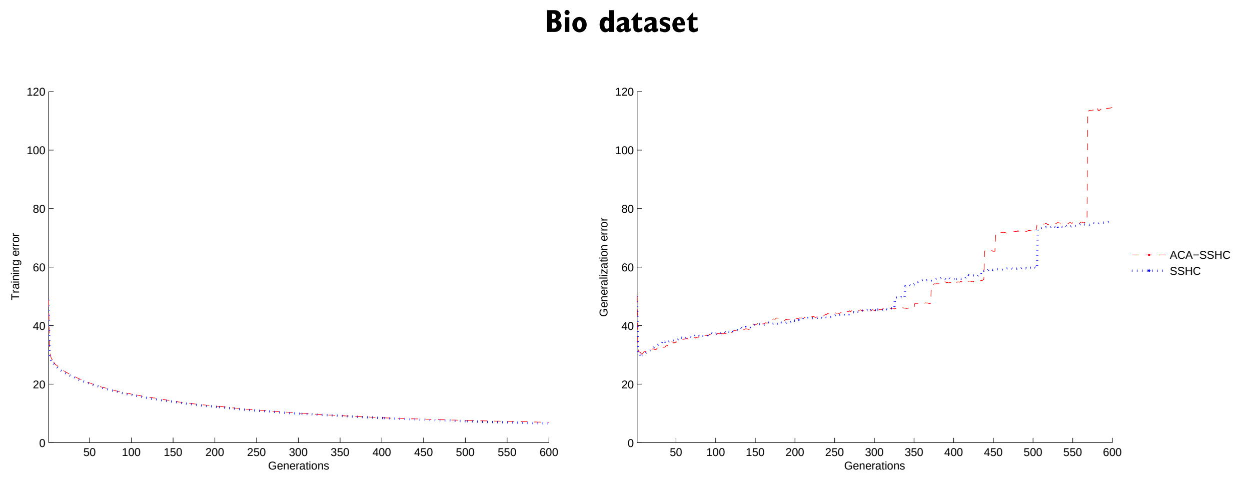
of overfitting is not particular to an approach with near-optimal alignments, and that is more clearly related to the very low training error thresholds defined. For the purpose of further comparison between SSHC and ACA-SSHC, and since there is no clear influence of k, the value 1.1 will be used for the remaining comparisons (as used before in figure 5.16). In terms of the number of generations needed to reach the threshold, SSHC is superior, with statistical significance (p-value 6.171×10^{-4}), to ACA-SSHC in the PPB dataset. There is no statistically significant difference for the number of generations in the other datasets.

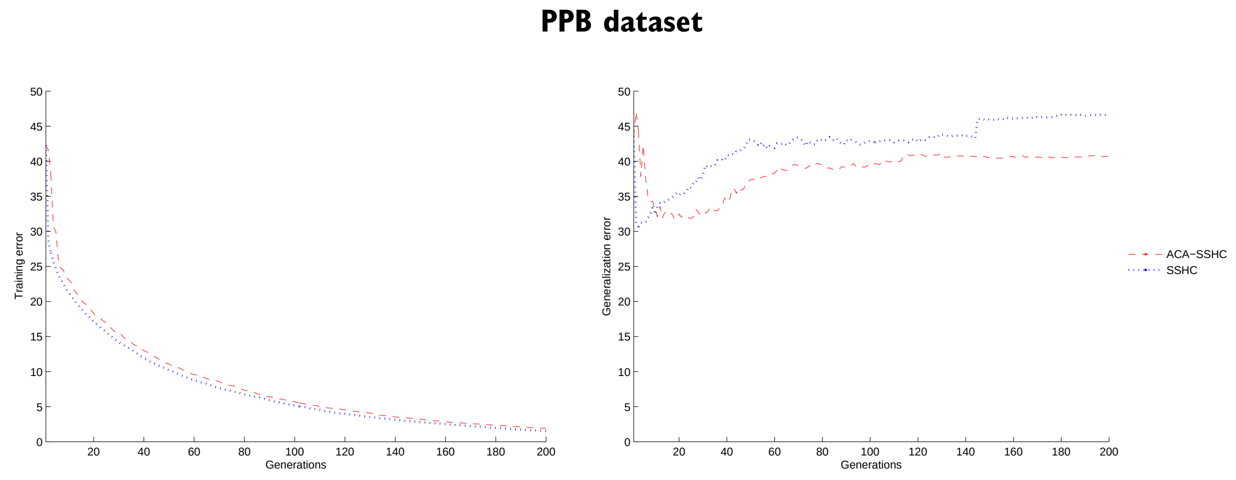
Figure 5.17 shows the training and generalization errors evolution for ACA-SSHC using $k = 1.1$ and SSHC. Generally, both methods present similar trends during the evolution. From the evolution plots and with the exception of the Bio dataset, it is clear that SSHC fits the training data faster in the beginning of the evolution. This is particularly the case in the LD50 dataset, although ACA-SSHC does reach a similar training error at around 400 generations. In the PPB dataset there is always a small gap between the two methods. As mentioned before, this then translates into SSHC reaching the stopping threshold faster than ACA-SSHC. These results suggest that directly minimizing the training error may be more efficient than searching for an alignment in the beginning of the evolution. However, it seems that there is no consistent difference for the total number of generations needed to reach similar training error thresholds. In terms of generalization and in all datasets, the two approaches start overfitting early on.

Table 5.8: Generalization error and number of generations taken to achieve the desired training error threshold for SSHC

Dataset	**Generalization error**			**Generations**		
	Median	**Average**	**SD**	**Median**	**Average**	**SD**
Bio	104.4	1.1e+08	5.2e+08	1099.0	1103.2	321.9
PPB	49.7	3.9e+09	2.1e+10	244.0	246.8	21.6
LD50	4457.4	3.8e+26	2.0e+27	2353.0	2440.7	485.3

An interesting result to notice is that it is possible, in some runs, to reach near-optimal alignments while still achieving a competitive generalization. Table 5.9 shows the generalization error for the best runs of each alignment experiment. In this context, the best runs are defined according to generalization error, i.e., the runs with lower generalization error. For the PPB dataset, there are 7 runs that present a competitive generalization





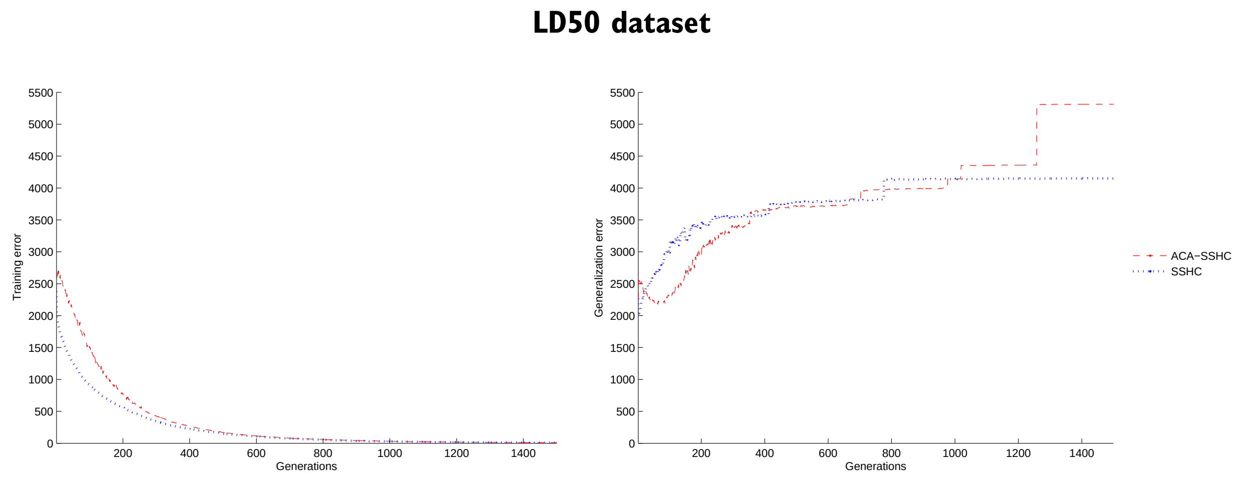


Figure 5.17: Training and generalization errors evolution plots for ACA-SSHC with $k = 1.1$ and SSHC

(around or lower 30 RMSE). Particularly interesting are the results for $k = -0.9$ and $k = 0.1$, respectively achieving 26.1 and 27.0 RMSE. However, in the Bio and LD50 datasets it was not possible to achieve a competitive generalization (around 30 and 2000 RMSE respectively) on any run. For the LD50 dataset, this could be because the alignments are even closer to optimality. This translates into relatively smaller training errors and it is consequently more prone to overfitting.

Although the overall results show that near-optimal alignments are risky in terms of overfitting, the best runs for the PPB dataset show that individuals constructed from near-optimal alignments can still generalize well. This refers back to the model (or individual) selection problem mentioned before. It is possible to construct individuals from near-optimal alignments that generalize well, but their selection is not likely. A possible research venue is to identify some additional criteria that could help identify whether a given alignment/run is likely to be competitive in terms of generalization.

Table 5.9: Generalization error of the best runs for ACA-SSHC

k	**Bio**	**PPB**	**LD50**
0.10	43.9	27.0	3074.8
-0.10	38.2	35.1	2837.1
0.25	39.3	32.4	2796.3
-0.25	36.4	32.0	2541.8
0.50	40.8	31.1	2903.5
-0.50	42.5	30.7	2965.6
0.75	47.7	30.5	2744.8
-0.75	44.4	31.8	2748.4
0.90	35.4	31.2	2879.8
-0.90	51.6	26.1	2969.7
1.10	41.2	27.7	2963.0
-1.10	38.4	29.2	3102.7
1.25	42.8	31.4	2926.0
-1.25	35.4	27.9	2890.0
1.50	39.6	28.9	2683.2
-1.50	40.7	30.8	2871.0
1.75	41.6	30.0	2604.0
-1.75	40.8	31.0	2893.0
2.00	44.0	30.8	2878.2
-2.00	37.1	29.8	2938.6

6 Semantic Learning Machine

This chapter proposes a feedforward Neural Network construction algorithm that shares the same geometric semantic properties of Geometric Semantic Genetic Programming. This construction algorithm is named Semantic Learning Machine (SLM). Section 6.1 defines the SLM. Section 6.2 empirically assesses the learning and generalization capabilities of the proposed algorithm. Section 6.3 studies the incorporation of semantic stopping criteria in the SLM and GSGP. Finally, section 6.4 provides a comparison with non-evolutionary supervised learning methods.

6.1 Definition

6.1.1 Overview of Artificial Neural Networks

Artificial Neural Networks (NNs) are computational models inspired by the functioning of the human brain. The problem solving approach underlying NNs is based on a relatively simple processing unit, which when replicated and combined with other similar units, results in a potentially complex behavior. This processing unit is inspired by the human neuron. It can generally be referred to as an artificial neuron, or more loosely as a perceptron. From now on, this processing unit is simply referred to as neuron. The role of each neuron is to take a set of inputs with a corresponding set of weights, and to perform a transformation of the data received. The result of this transformation is then passed on to the next neurons. Formally, let $\overrightarrow{x}$ be the vector containing the inputs passed to a given neuron, and $\overrightarrow{w}$ the vector containing the corresponding weights of each input. Both vectors have the same number of elements, so that w_i is the weight applied to input x_i. The output (y) of a neuron is computed as follows:

$$y = f(\overrightarrow{x} \times \overrightarrow{w} + bias), \tag{6.1}$$

where $\times$ represents the scalar product between two vectors, and f is commonly a non-linear function, usually referred to as a transfer or activation function. A bias is added to each neuron. Figure 6.1 exemplifies a neuron.

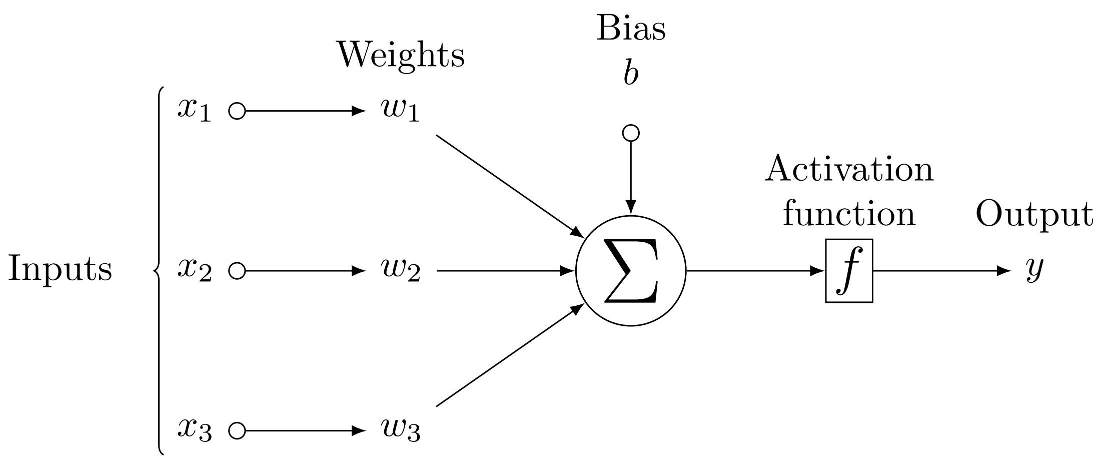


Figure 6.1: An artificial neuron with three inputs

These neurons are combined in layers to form a network. This network is a type of neural network commonly known as a Multilayer Perceptron (MLP). A layer is defined by a set of elements that are not connected with each other, but are connected with all the elements of the next layer. This is known as a fully connected MLP. A MLP consists of three types of layers: input, hidden, and output. The flow of data processing is always carried on from the input layer to the output layer. This is known as a feedforward NN. For the remainder of this chapter, the term NN is used to describe a feedforward MLP. Figure 6.2 exemplifies a NN. Since there are no backward connections, any feedforward NN can be represented by a tree. This means that given suitable function and terminal sets, GP can represent any feedforward NN. A NN with backward connections is a recurrent NN. Given that this type of NNs creates cycles within its network, it can not be directly represented by a tree. In this chapter only feedforward NNs are considered. In these

NNs, there are always only one input and output layers, but there can be more than one hidden layer. The hidden layers and the output layer are composed of the neurons previously described. The input layer consists of simpler elements that just take the input provided and pass it on to the elements of the next layer. The elements of the input layer are equivalent to the terminal nodes in GP. The activation functions of the neurons of the hidden layers are commonly sigmoidal functions, such as the logistic function used in GSGP in the previous chapter. It is usual for all the neurons in a given hidden layer to use the same activation function. In the simpler case the output layer can consist of a single element. Without loss of generality, this is the case considered here. For the output layer the activation function considered is the identity function:

$$f(x) = x \tag{6.2}$$

This is a commonly used output layer activation function when dealing with regression tasks.

Given that the geometric semantic operators are defined over the semantic space (outputs), they can be extended for different computational models or representations. In the next subsections a geometric semantic mutation for NNs is proposed, and a subsequent construction algorithm is defined. This allows an effective stochastic search to be performed in the space of NNs.

6.1.2 A Geometric Semantic Mutation Operator for Feedforward Neural Networks

As shown in the previous chapter, the geometric semantic operators are relatively simple combinations of trees. Particularly, the geometric semantic mutation (from definition 2) is a linear combination of two trees: the tree of the parent and a randomly generated tree (which results from subtracting the two random subtrees T_{R1} and T_{R2}). There are no restrictions on the two trees being combined. The crucial aspect is that these two trees are combined independently, i.e., the semantics of one the trees does not affect the semantics of the other tree, and vice versa. Generalizing, this particular geometric semantic mutation is defined as a linear independent combination of two computational elements. This knowledge can be used to create equivalent geometric semantic mutations for other

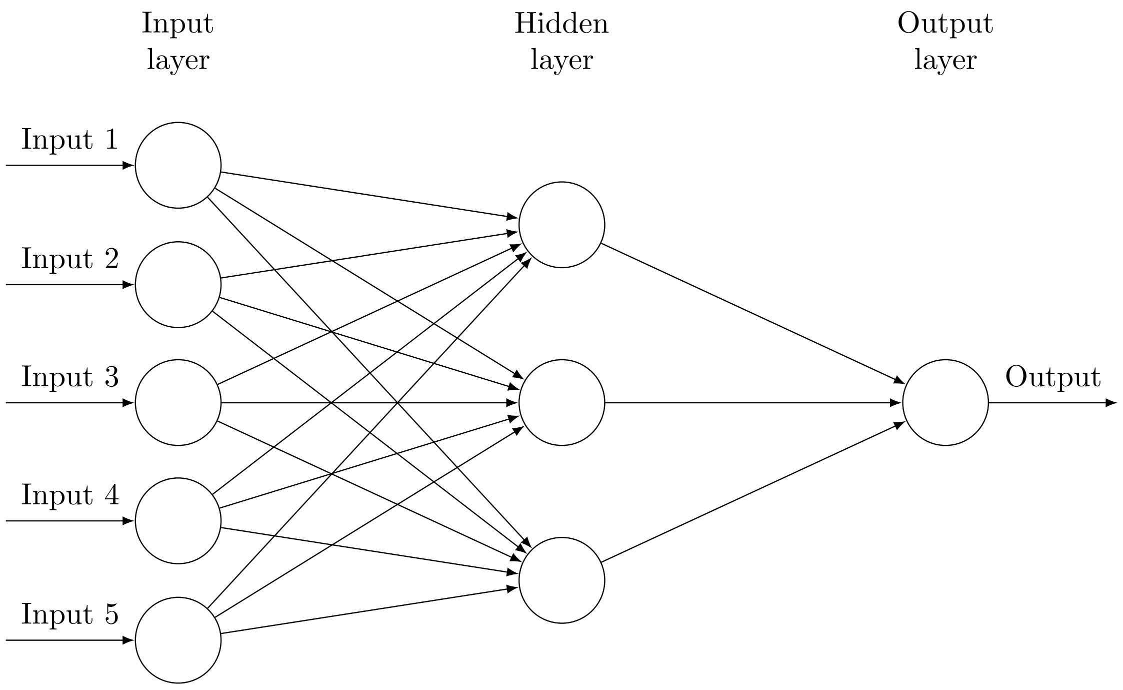


Figure 6.2: A feedforward neural network with five input neurons, a hidden layer of three neurons, and one output neuron

computational models or representations. To simplify, the proposed geometric semantic mutation for NNs is referred to as GSM-NN, from Geometric Semantic Mutation - Neural Networks. GSM-NN works by taking an already existing NN, and combining it linearly with a randomly generated NN. Analogously to the GSGP mutation, the already existing NN is equivalent to the parent tree, and the randomly generated NN is equivalent to the randomly generated tree. The already existing NN is referred to as the parent NN. As mentioned previously, the case being considered here is that of an output layer with a single neuron. To provide a rigorous analogy to the GSGP mutation, the activation function of the output neuron is the identity function. For the same reason, the bias is excluded from the output neuron. The following description of GSM-NN assumes a single hidden layer. This is also the mutation version that is experimentally tested in this chapter. A description of GSM-NN for multiple hidden layers will also be presented.

Each GSM-NN application consists of two main steps: creating a randomly generated NN, and joining this NN with the parent NN. Note that the simplest case of a NN is a NN with a single hidden layer neuron. This is the type of random NN that GSM-NN creates. Since joining two NNs implies sharing the input and the output layers, the only neuron that needs to be created in GSM-NN is the neuron to be added to the hidden layer. The weight from the new neuron to the output neuron is the learning step (a SLM parameter). This learning step is equivalent to the mutation step in the GSGP mutation. It influences the amount of semantic variation for each application of the operator. Similarly to GSGP, this learning step can be fixed throughout the run, or it can be computed optimally for each application of the operator. The fixed learning step version will be referred to as FLS, and the optimal learning step version as OLS. The weights from the input layer to the hidden layer are randomly generated. This is the equivalent of generating the random tree in the GSGP mutation. There are no restrictions on how the weights are generated. In this work these weights are generated between -1.0 and 1.0 with uniform probability. The activation function of the new neuron can be freely chosen. However, it should be a non-linear function in order to learn possible non-linearities in the data. As seen in the previous chapter, a positive effect on the generalization ability can be achieved by using a function with a relatively small codomain, and at the same time using a small learning/mutation step. To ensure a fair comparison with the GSGP mutation, the activation function used in GSM-NN is the hyperbolic tangent. This guarantees that, for each application of the operator, the semantic variation always ranges in the interval $[-ls, ls]$, where ls represents the learning step. This results from the fact that the outputs of the hyperbolic tangent range in the interval $[-1, 1]$. Similarly, in the GSGP bounded mutation the semantic variation always ranges in the interval $[-ms, ms]$, where ms represents the mutation step. As previously mentioned, the learning step and the mutation step are equivalent parameters. Figure 6.3 exemplifies the GSM-NN operator for the single hidden layer case.

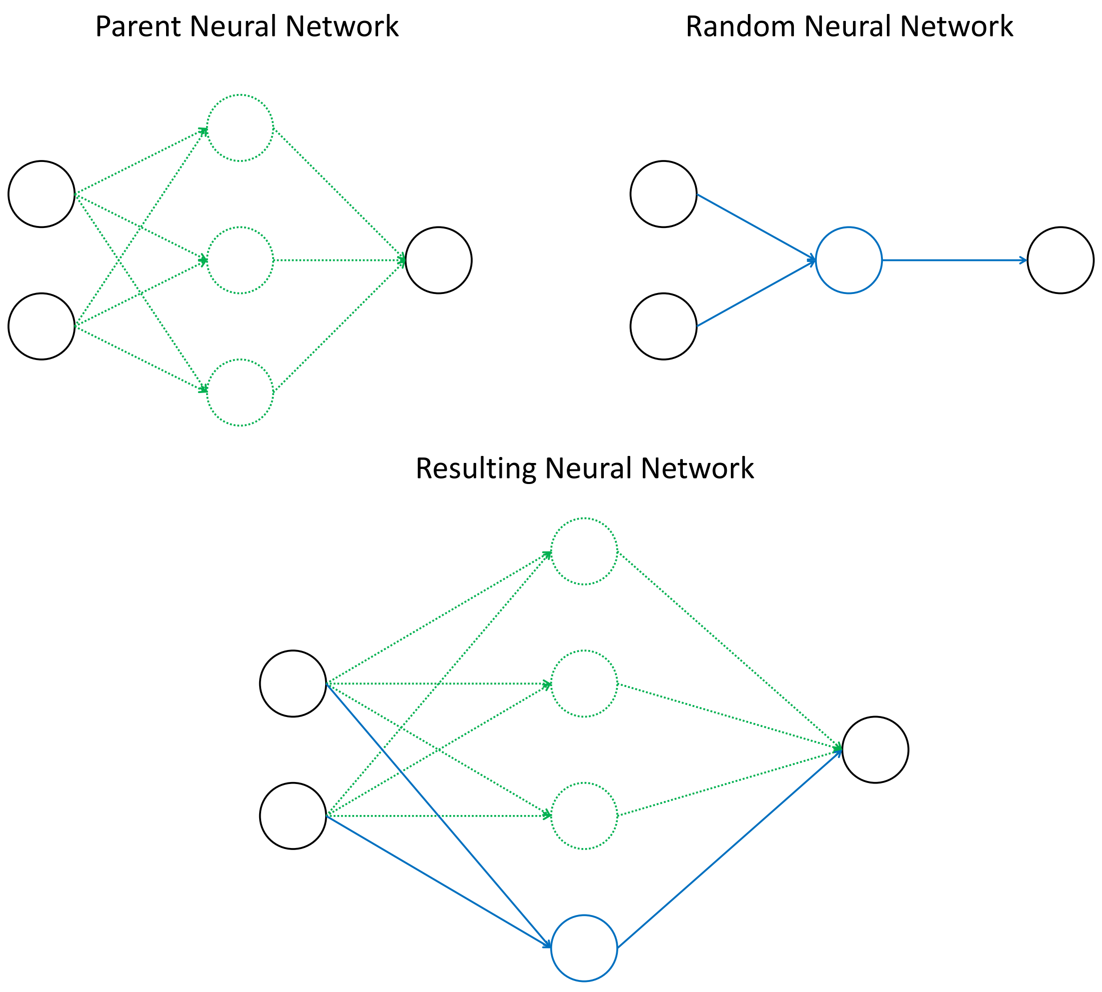


Figure 6.3: An example of an application of the GSM-NN operator

Similarly to the GSGP mutation, GSM-NN induces an unimodal fitness landscape with a constant slope in the space of NNs. This is valid for all supervised learning problems. Consequently, GSM-NN is potentially very effective in terms of search since the type of landscape being searched on has no local optima. This is experimentally assessed in the next section. Notice that this type of landscape has clear consequences for the learnability of NNs. This contrasts with the universal approximation results (Hornik et al., 1989) that deal with the representational issue. These results establish that feedforward NNs with even a single hidden layer can approximate any measurable function to any desired degree of accuracy, given a sufficient but finite number of hidden layer neurons. However, the universal approximation results (Hornik et al., 1989) leave open the question of learnability of the feedforward NNs.

The working of GSM-NN for multiple hidden layers is similar to the single hidden layer case. A new neuron is added to the last hidden layer, and the weight that connects this neuron with the output neuron is the learning step. There are different valid ways of defining how the connections up to the new neuron are created. The simplest case just involves generating random weights from the neurons of the penultimate hidden layer to the new neuron. In this case all other weights remain unchanged. Other possibilities involve relaxing the property of full connectivity of the resulting NN. One of the most complex cases would involve adding one neuron to each hidden layer. Each new neuron could be fully connected to the previous hidden layer, but in terms of forward connections it could only connect to the new neuron of the next hidden layer. Constructing a fully connected network in this case would result in a non-independent combination in the output neuron, and would therefore not be a valid geometric semantic mutation. Given the different possibilities when using the multiple hidden layer GSM-NN, this chapter only explores experimentally the single hidden layer GSM-NN. A geometric semantic crossover for NNs would also be possible to formulate. However, it would share similar disadvantages to the GSGP crossover, particularly the issue of the exponential growth of the solutions. Furthermore, no a priori advantages in terms of search are known for the geometric semantic crossover that could outweigh its disadvantages. For these reasons, the geometric semantic crossover for NNs is not explored.

6.1.3 Construction Algorithm

The SLM algorithm is essentially a geometric semantic hill climber for NNs. Since the semantic space has no local optima, the search can be focused around the current best NN without incurring in any particular disadvantage. As seen in the previous chapter, using this strategy is more efficient than using a population of solutions. The SLM algorithm starts by generating a set of random NNs. The number of random NNs is defined by the sample size SLM parameter (*N*). From these initial NNs, the best one is selected according to the training error. After this initial step, the main part of the hill climbing begins. A set of new NNs is produced by applying the GSM-NN operator to the current best NN. The number of new NNs produced is also defined by the sample size parameter. From these NNs, the overall best NN is updated. The process is repeated for a given number of iterations (SLM parameter). The number of iterations (or epochs) parameter is equivalent to the number of generations parameter in GSGP/SSHC. Alternatively, a different stopping criterion can also be used to determine when to stop the search (this is explored in section 6.3). The algorithm is summarized in the following steps:

1. Generate *N* initial random NNs
2. Choose the best NN (*B*) from the initial random NNs, according to the training error
3. Repeat the following operations until a given stopping criterion is met:
    3.1. Apply GSM-NN to the current best (*B*) *N* times to generate *N* new NNs
    3.2. Update *B* as being the NN with the smallest training error, selected from the current *B* and the *N* newly generated NNs
4. Return *B* as the best performing NN according to the training error

The initial random NNs are generated similarly to the random NNs used in the GSM-NN operator. These random NNs are generated with a single hidden layer and a single neuron in the hidden layer. The weights from the input layer to the hidden layer are generated randomly between -1.0 and 1.0 with uniform probability. The weight from the hidden layer neuron to the output neuron is set to 1. This weight can also be computed optimally with the Moore-Penrose pseudoinverse, as seen in the previous chapter. Other ways of generating these initial random NNs could be considered.

It is important to remark that the GSM-NN operator excludes the need to use backpropagation to adjust the weights of the network. The use of backpropagation entails two considerable disadvantages. The first disadvantage is that, in general, the space of the weights of a NN is not unimodal in relation to the overall error. In other words, a set of weights can be locally optimal, but not globally optimal. This means that the search over the space of weights needs to take into consideration the issue of local optima. In practice, a type of search that needs to circumvent this issue is commonly less effective and efficient. As mentioned, the SLM search operates over a unimodal space, which can theoretically simplify the search. This is assessed empirically in the next sections. The second disadvantage of using backpropagation is the computational cost. As the networks grow in size, backpropagation starts to become considerably slow. By excluding to adjust the weights, the GSM-NN operator is able to keep the same computational cost as the network grows. This is possible because GSM-NN can always perform an incremental evaluation. In other words, regardless of the size of a NN, the evaluation only needs to occur in the new part of the NN. Considering the single hidden layer case, GSM-NN only needs to evaluate the contribution of the neuron that is being added to the NN. The contribution of the rest of the NN is already computed, and therefore does not need to be reevaluated. This makes the SLM algorithm very efficient in practice.

6.2 Experimental Study

This section provides an empirical assessment of the learning and generalization capabilities of the SLM. The experimental setup is equivalent to the one used in the previous chapter. This allows a direct comparability with the results presented in the previous chapter.

6.2.1 Initial Exploration

This subsection studies the effects of the learning step and the sample size in the SLM when using a fixed learning step.

The values tested for the learning step (LS) are: 1, 10, and 100. Figure 6.4 presents the evolution plots with the training and generalization errors for the different learning steps tested. In the Bio and PPB datasets, a LS of 1 is found to be the best suited for both training and generalization errors. LS 10 and 100, end up being less effective. This implies that the semantic adjustments (influenced by the LS) become too high at some point during the search. In the LD50 dataset, all learning steps provide an effective training error reduction. This difference to the other two datasets is explained by the higher order of magnitude of the errors in the LD50 dataset. Since these errors are bigger, it should be expected that higher learning steps are needed (at least in the initial search phases). LS 100 achieves the best training error but compromises the generalization ability. Given this fact, LS 10 ends up being the best trade-off in terms of learning and generalization. Overall, the results are consistent with the established notions of the effects that different learning/mutation steps have in search. Given these results, for the remainder of this chapter, a LS of 1 is used in the Bio and PPB datasets, and a LS of 10 is used in the LD50 dataset.

The values tested for the sample size (SS) are: 10, 50, 100, and 500. Figure 6.5 presents the evolution plots with the training and generalization errors for the different sample sizes tested. As a general and expected trend, the bigger the SS, the easier it is to decrease the training error. This is natural given that more resources are applied in the search process. In terms of generalization, the values achieved are similar regardless of the SS used. It seems that the tendency to overfit is more clearly related to the number of iterations/generations conducted, than to the sample/population size. Across all datasets, results show that even a small SS (10) is able to effectively search the semantic space and achieve a competitive generalization. Regardless, to maintain a direct comparability with the GSGP/SSHC results presented in the previous chapter, a SS of 100 is used in all datasets for the remainder of this chapter.

6.2.2 Sparseness

The concept of sparseness has different interpretations depending on the context. Here, sparseness is used to refer to the number of input variables effectively used in a given model. As sparseness increases, the number of input variables effectively used decreases. In the context of the NNs considered here, the sparseness percentage defines the percentage of weights from the input layer to the hidden layer that are exactly zero. For instance, a sparseness of 95% implies that only 5% of the input variables are used in a NN, i.e., only 5% of the mentioned weights are different than zero. A sparse model is structurally simpler, and could potentially generalize better. In the SLM, sparseness is easily controlled in the GSM-NN operator. Since GSM-NN creates a random NN in each operator application, the sparseness can be directly controlled by defining the num-

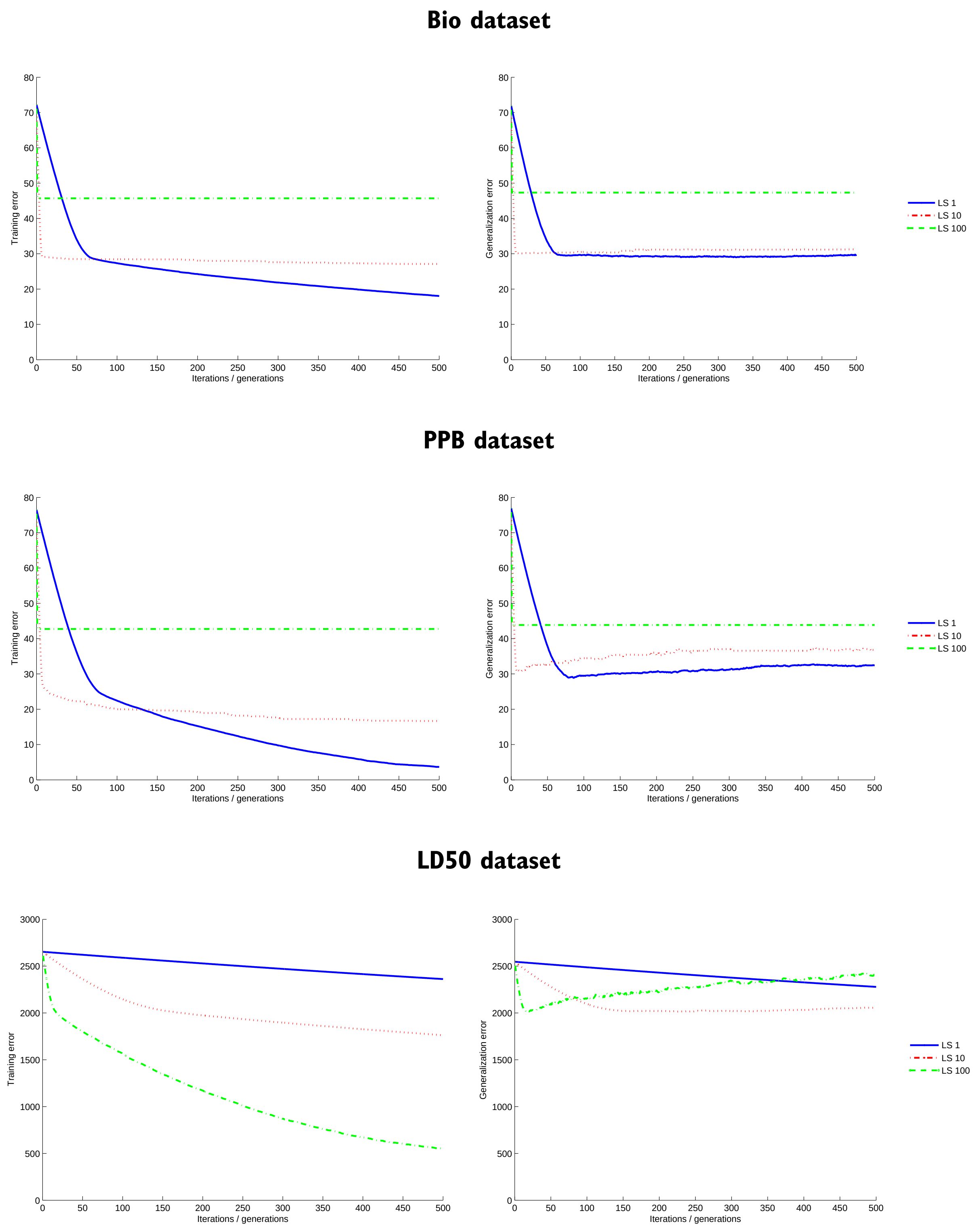


Figure 6.4: Training and generalization errors evolution plots for SLM with different learning steps

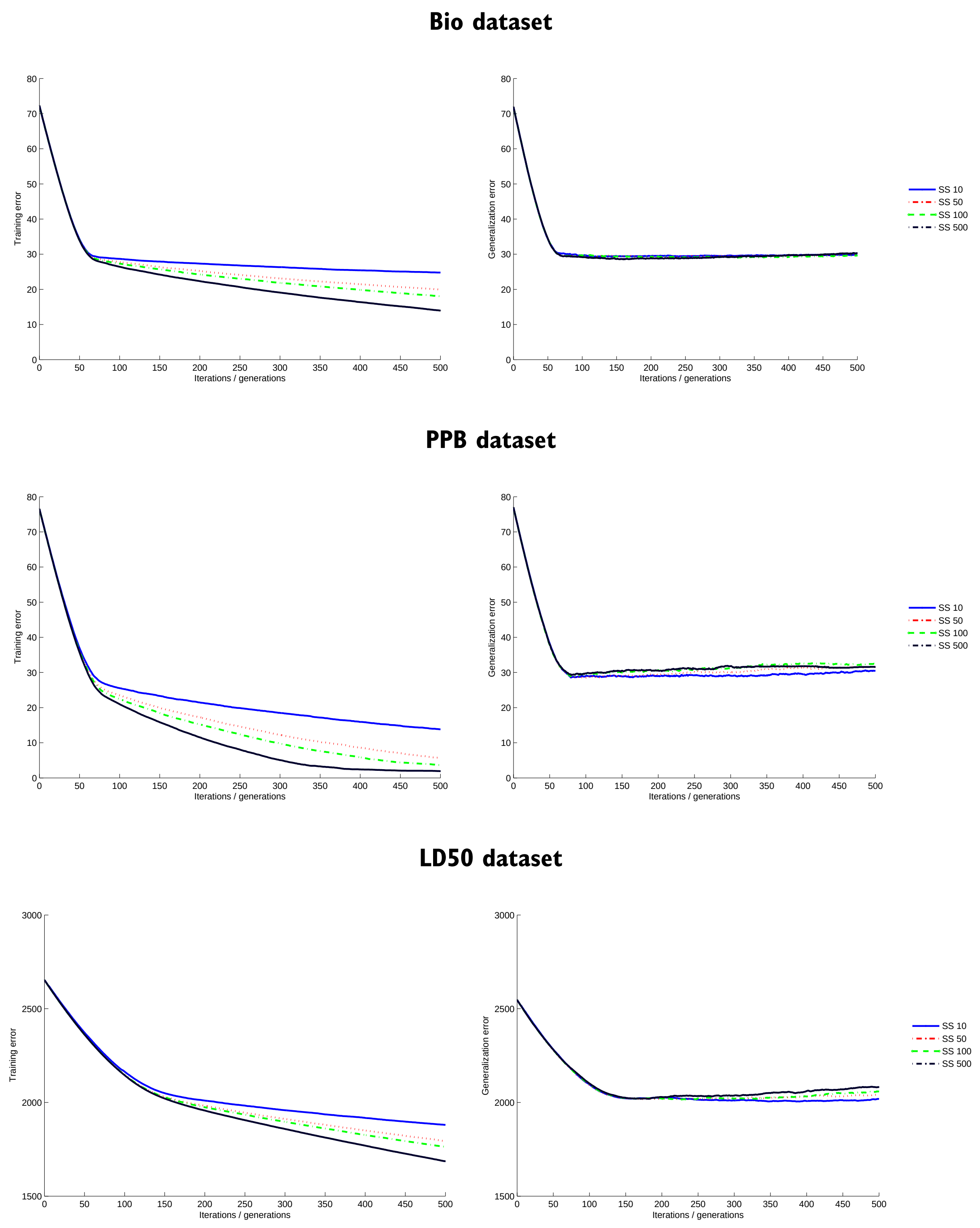


Figure 6.5: Training and generalization errors evolution plots for SLM with different sample sizes

ber of weights that are going to be set to zero. The weights that are going to be set to zero are selected randomly. The other weights can be generated as usual. Therefore, sparseness is controlled for each application of the operator, and not for the final NN. In other words, even if all applications of GSM-NN use the same given percentage of sparseness, it is not guaranteed that the final NN will only use the same corresponding number of input variables. Since each GSM-NN application randomly selects the input variables that are going to be used, different GSM-NN applications may select different input variables, which leads to the final NN using more than the input variables defined for a single GSM-NN application. Table 6.1 presents the correspondence between the sparseness percentage and the number of input variables used in each dataset considered.

Table 6.1: Correspondence between the sparseness percentage and the number of input variables

Dataset	**Sparseness**	**Corresponding number of input variables**
Bio	0%	241
	5%	229
	25%	181
	50%	120
	75%	60
	95%	12
PPB	0%	627
	5%	596
	25%	470
	50%	313
	75%	157
	95%	31
LD50	0%	626
	5%	595
	25%	469
	50%	313
	75%	156
	95%	31

The values tested for the sparseness percentage (SP) are: 0%, 5%, 25%, 50%, 75%, and 95%. Figure 6.6 presents the evolution plots with the training and generalization errors for the variants tested. As in the previous subsection, the SLM is tested using a fixed learning step. In addition to the evolution plots, the boxplots with the generalization errors achieved at the end of the runs are presented in figure 6.7. SLM presents robustness to all levels of sparseness tested as all variants present similar generalization behaviors. In general, increasing the sparseness level does not increase the generalization achieved. The only exception occurs in the PPB dataset, where using a sparseness level of 95% improves the generalization achieved (p-value $\mathbf{1.905} \times \mathbf{10^{-3}}$). A bigger effect is detected in terms of training error. In the Bio dataset, using any sparseness level improves significantly the training performance: 5% (p-value $\mathbf{3.759} \times \mathbf{10^{-3}}$), 25% ($p$-value $\mathbf{7.044} \times \mathbf{10^{-10}}$), 50% ($p$-value $\mathbf{3.175} \times \mathbf{10^{-11}}$), 75% ($p$-value $\mathbf{2.872} \times \mathbf{10^{-11}}$), and 95% ($p$-value $\mathbf{3.510} \times \mathbf{10^{-11}}$). In the PPB dataset, a sparseness level of 50% also improves the training performance (p-value $\mathbf{3.878} \times \mathbf{10^{-4}}$). No training performance advantages are found in the LD50 dataset.

6.2.3 Fixed Learning Step Comparisons

This subsection presents a comparison between the SLM with a fixed learning step (SLM-FLS) and SSHC BM. SSHC BM is the tree-based method equivalent to SLM-FLS. They both use fixed learning/mutation steps, and produce semantic variations in the same range ($[-ls, ls]$, where ls is the learning/mutation step) at each mutation application. For reference, the results for SLM-FLS with high sparseness levels are also presented: with 75% sparseness (SLM-FLS SP 75%), and with 95% sparseness (SLM-FLS SP 95%). Figure 6.8 presents the evolution plots with the training and generalization errors for the methods tested. The first point to notice are the considerably different values at iteration/generation 0. SLM-FLS presents much higher errors than SSHC BM after the random initialization. This is influenced by the fact that the initial weights in SLM-FLS are generated with uniform probability between -1.0 and 1.0. This effectively bounds the amount of fitting that can be achieved in the initialization. On the other hand, SSHC BM has no explicit bound on the initial random trees, and can therefore provide a superior initial explanation of the data. It is interesting to note that, despite this initial disadvantage, SLM-FLS compensates with a considerably higher learning rate. This translates into a statistically significant superiority in terms of training error across all datasets (p-values: Bio $\mathbf{2.872} \times \mathbf{10^{-11}}$, PPB $\mathbf{2.872} \times \mathbf{10^{-11}}$, and LD50 $\mathbf{3.261} \times \mathbf{10^{-5}}$). This learning superiority is particularly interesting when considering that SLM-FLS and SSHC BM use equivalent geometric semantic mutation operators. The differences arise from the different initial-

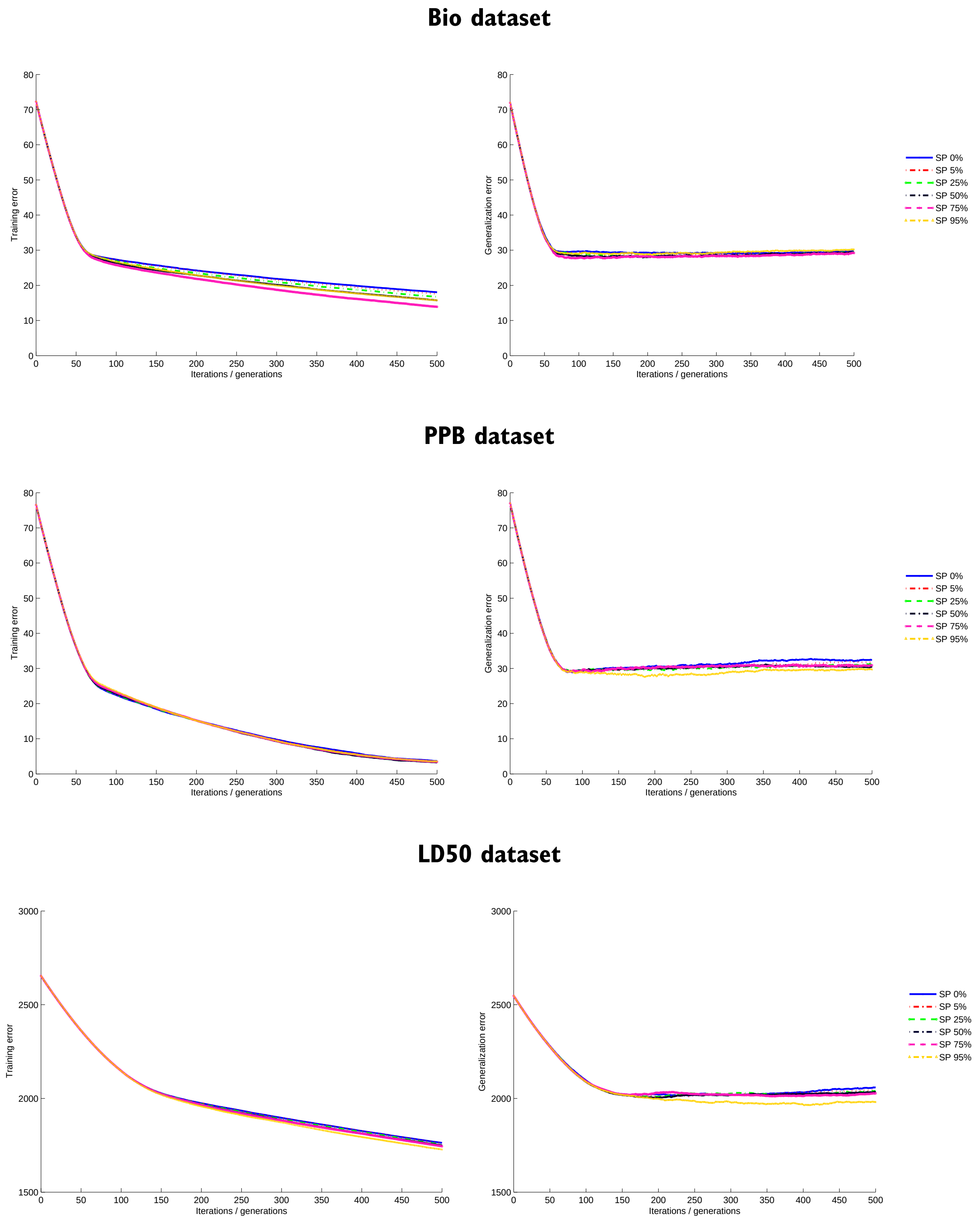


Figure 6.6: Training and generalization errors evolution plots for SLM with and without sparseness

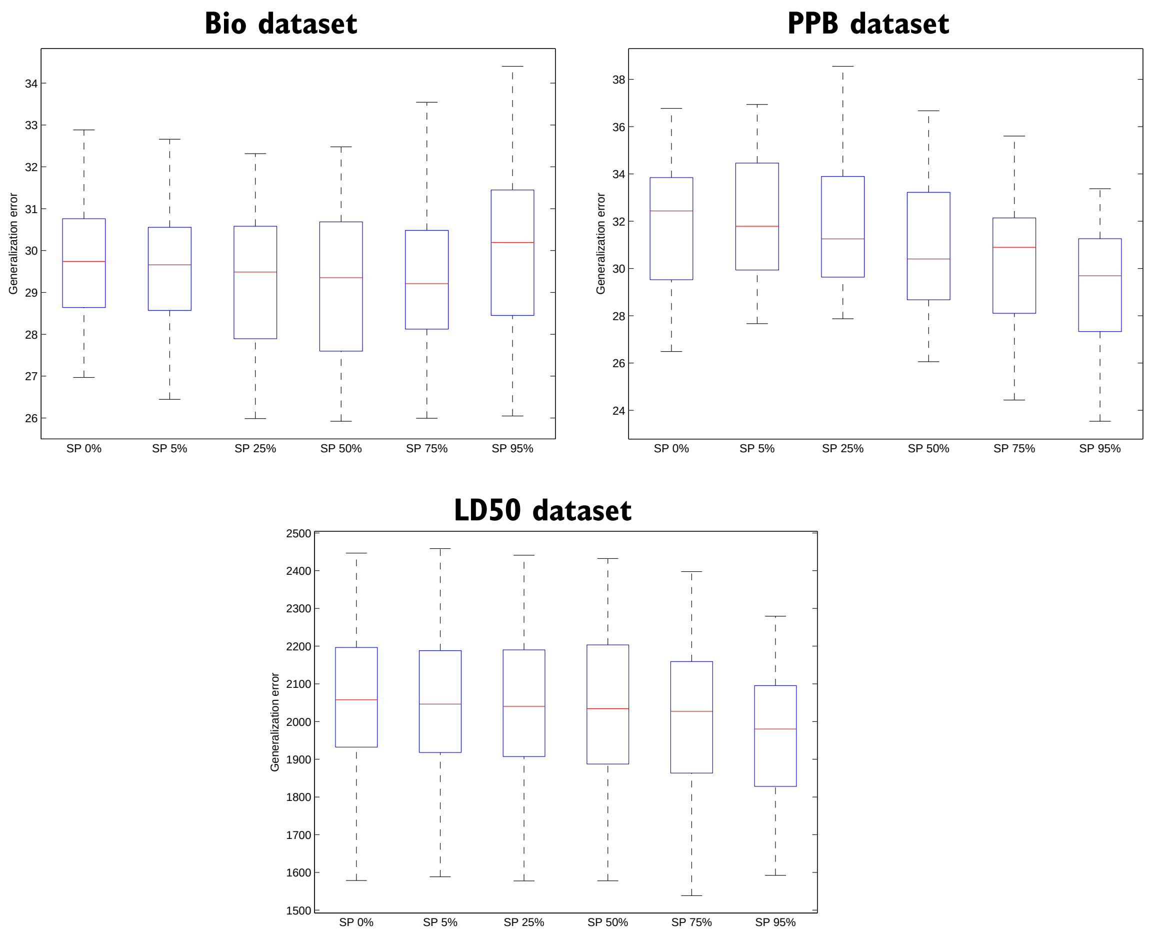


Figure 6.7: Generalization error boxplots for SLM with and without sparseness

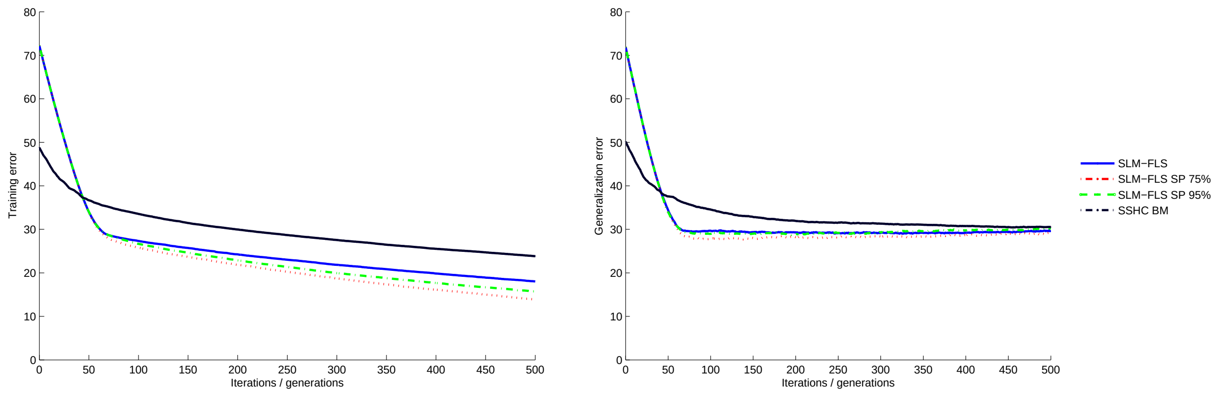
izations used in both representations (NNs and trees). Different representations have different natural ways of being randomly initialized. These different initializations result in different semantic distributions, which in turn lead to different offspring distributions. At least in these datasets, the semantic distribution resulting from the random initialization of a list of weights (used in SLM-FLS) is more favorable than the semantic distribution resulting from the random tree initialization (used in SSHC BM). This first distribution can be loosely described as more well-behaved. In the original GSGP proposal, Moraglio et al. (2012) provided a discussion on whether syntax (representation) matters in terms of search. They argued that, in abstract, the offspring distributions may be affected by the different syntax initializations. From the SLM-FLS and SSHC BM results, it can be empirically seen how initializations over different representations can lead to different offspring distributions and to different learning outcomes. A possible research venue in GSGP lies in analyzing the semantic distributions produced by different tree initializations, and to propose new tree initializations that are more well-behaved.

In terms of generalization, results show that all methods achieve similar results as no statistically significant differences are found. However, SLM-FLS has the advantage of converging faster to the generalization plateau. The mentioned faster learning rate also applies to the generalization. This faster convergence can be exploited by using some criterion to stop the search as the generalization plateau is reached. Some possible stopping criteria are explored in the next section. Figure 6.9 presents the boxplots with the generalization errors achieved at the end of the runs. These boxplots confirm the consistency of the generalization achieved by SLM-FLS. SSHC BM achieves a similar generalization but some negative outliers exist.

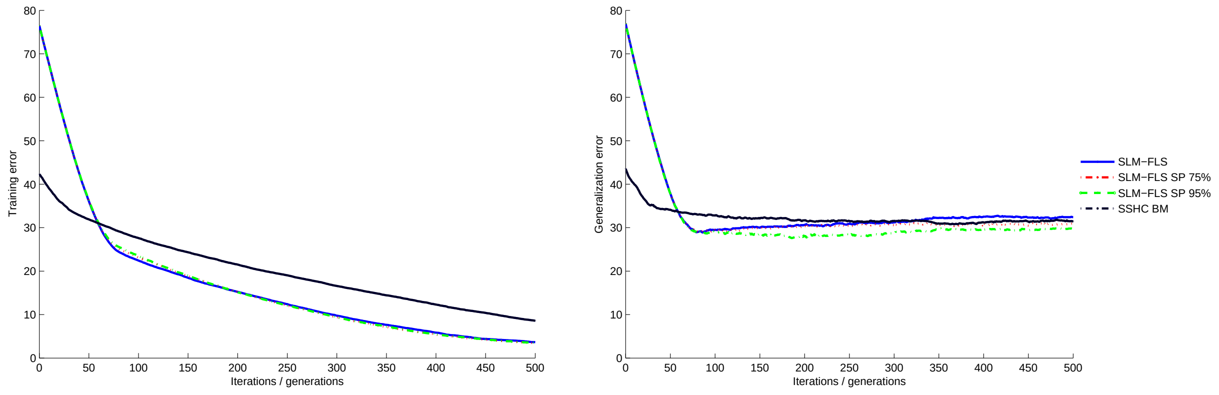
6.2.4 Optimal Learning Step Comparisons

This subsection presents a comparison between the SLM with an optimal learning step (SLM-OLS) and SSHC ABM. SSHC ABM is the tree-based method equivalent to SLM-OLS. They both compute optimal learning/mutation steps at each mutation application. Results for SLM-OLS with different sparseness levels are also presented. Figure 6.10 presents the evolution plots with the training and generalization errors for the methods tested. Both SLM-OLS and SSHC ABM fit the training data effectively. The computation of the optimal learning step allows SLM-OLS to quickly compensate the higher initial er-

Bio dataset



PPB dataset



LD50 dataset

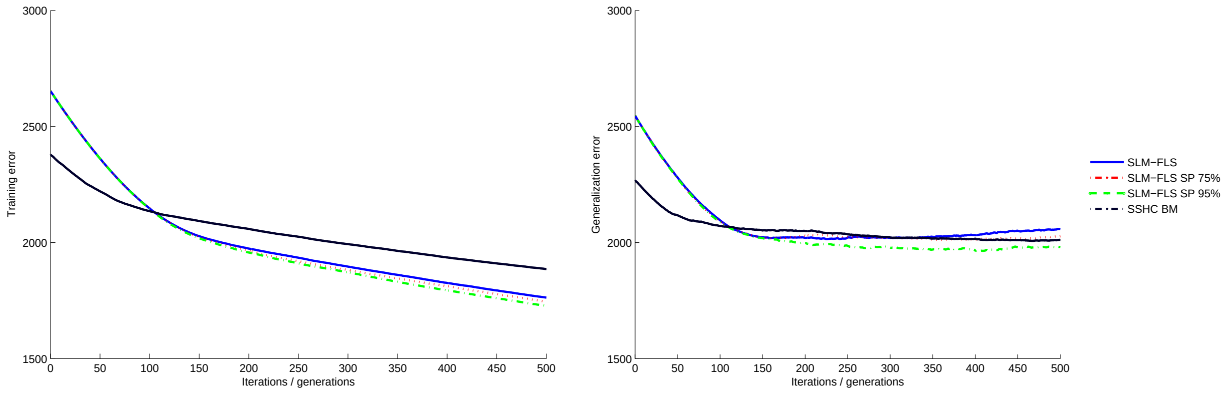


Figure 6.8: Training and generalization errors evolution plots for SLM-FLS, SLM-FLS SP 75%, SLM-FLS SP 95%, and SSHC BM

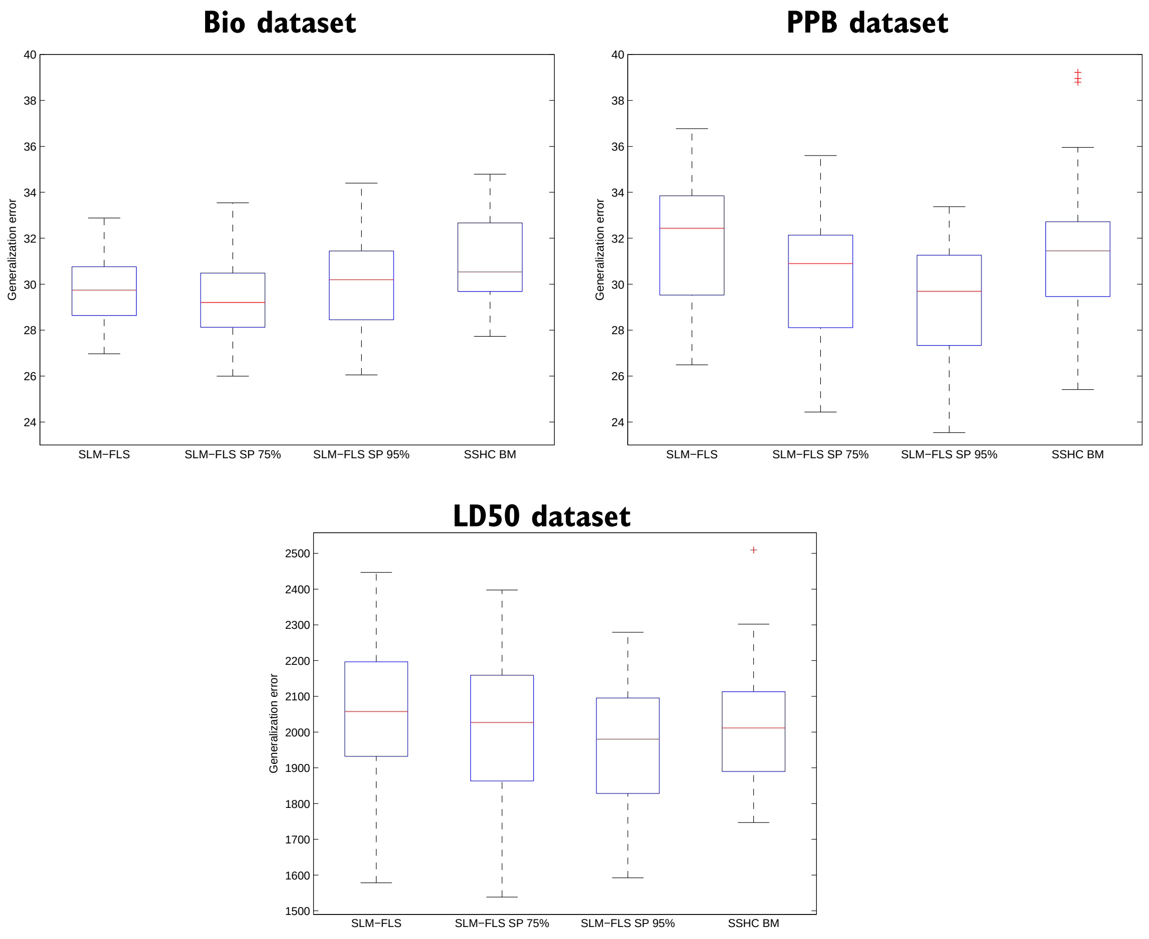


Figure 6.9: Generalization error boxplots for SLM-FLS, SLM-FLS SP 75%, SLM-FLS SP 95%, and SSHC BM

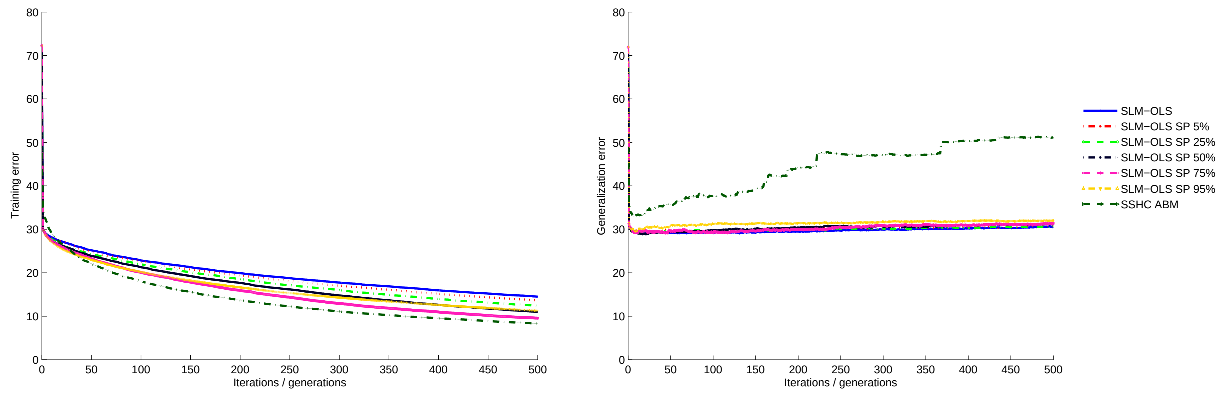
rors mentioned in the previous subsection. Therefore, SLM-OLS and SSHC ABM achieve similar training errors in the initial iterations/generations. After a few iterations/generations SSHC ABM starts to fit the training data faster, but this comes at the expense of overfitting. It is interesting to note that SSHC ABM only starts to fit the training data faster when no further generalization can be achieved. By the end of the runs, SSHC ABM is superior, with statistical significance, in terms of training error across all datasets (p-values: Bio $\mathbf{2.872} \times \mathbf{10^{-11}}$, PPB $\mathbf{2.872} \times \mathbf{10^{-11}}$, and LD50 $\mathbf{4.734} \times \mathbf{10^{-11}}$). On the other hand, SLM-OLS achieves a statistically significant superiority in terms of generalization across all datasets (p-values: Bio $\mathbf{4.734} \times \mathbf{10^{-11}}$, PPB $\mathbf{3.057} \times \mathbf{10^{-5}}$, and LD50 $\mathbf{1.800} \times \mathbf{10^{-7}}$). In terms of fitting the training data, it is also interesting to note that the computation of the optimal mutation step allows SSHC ABM to counteract the less well-behaved semantic distribution mentioned in the previous subsection.

In the Bio and PPB datasets, SLM-OLS is able to stabilize the generalization error even as the training error continues to be reduced. In the LD50 dataset, the generalization error increases slowly as the training error converges to zero. However, this increase is smaller than the one that occurs in SSHC ABM. Figure 6.11 presents the boxplots with the generalization errors achieved at the end of the runs. SSHC ABM is not shown as several values are of considerable order of magnitude, which makes these results very difficult to read and compare. SLM-OLS shows robustness to increasing levels of sparseness as all variants tested present similar generalization behaviors. In terms of generalization, no statistically significant differences are found at any of the sparseness levels considered. However, adding sparseness can be beneficial in terms of training error. In the Bio dataset, all variants with sparseness achieve significantly superior training performance than the variant without sparseness: 5% (p-value $\mathbf{8.401} \times \mathbf{10^{-5}}$), 25% ($p$-value $\mathbf{2.872} \times \mathbf{10^{-11}}$), 50% ($p$-value $\mathbf{2.872} \times \mathbf{10^{-11}}$), 75% ($p$-value $\mathbf{2.872} \times \mathbf{10^{-11}}$), and 95% ($p$-value $\mathbf{2.872} \times \mathbf{10^{-11}}$). In the PPB dataset, similar superiorities are achieved for sparseness levels of 25% or above: 25% (p-value $\mathbf{5.411} \times \mathbf{10^{-4}}$), 50% ($p$-value $\mathbf{2.063} \times \mathbf{10^{-5}}$), 75% ($p$-value $\mathbf{2.320} \times \mathbf{10^{-4}}$), and 95% ($p$-value $\mathbf{1.931} \times \mathbf{10^{-8}}$). No advantages are found in the LD50 dataset.

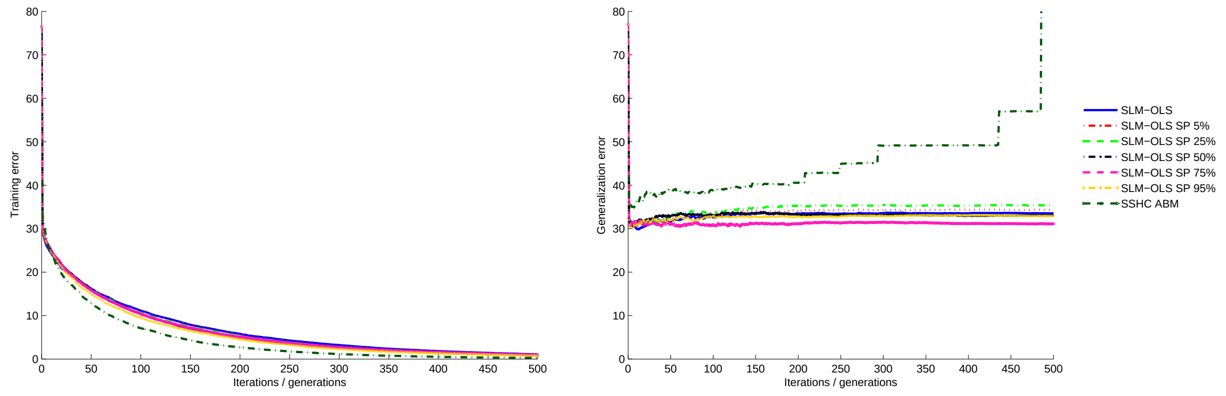
6.2.5 Normal Weight Distribution

Given the previously noted effect of different semantic distributions, this subsection explores how the use of a different weight distribution in SLM can influence its performance.

Bio dataset



PPB dataset



LD50 dataset

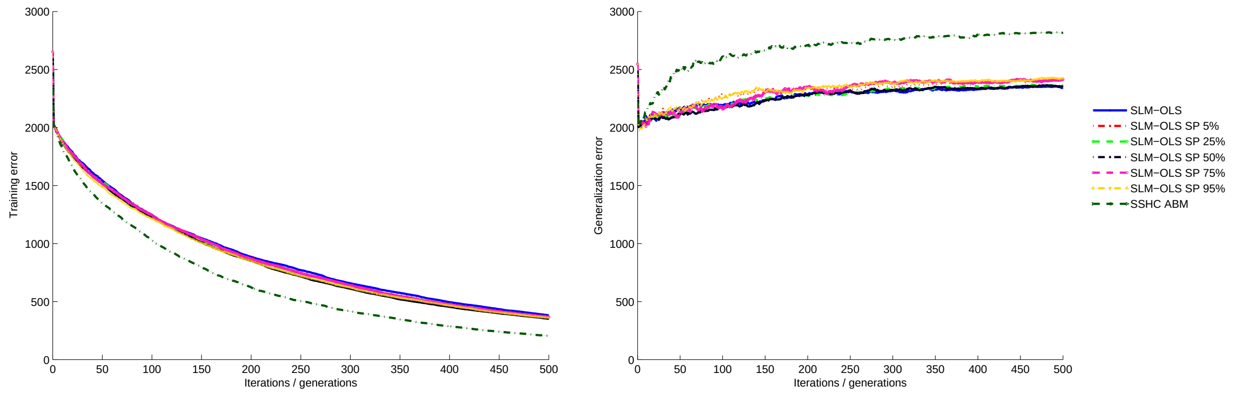


Figure 6.10: Training and generalization errors evolution plots for SLM-OLS with and without sparseness, and SSHC ABM

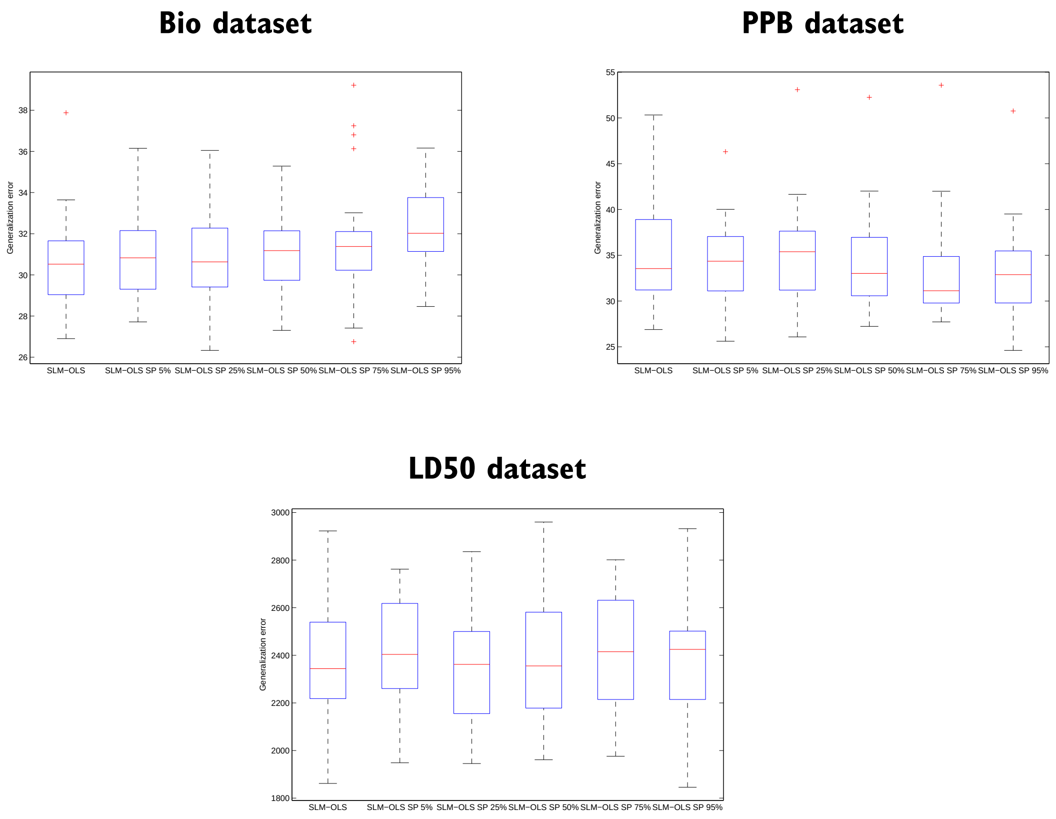


Figure 6.11: Generalization error boxplots for SLM-OLS with and without sparseness

Note that a different weight distribution influences the semantic distribution, which in turn influences the offspring distribution. The intent of this subsection is to provide an exploratory investigation by studying a different weight distribution. An exhaustive study of different weight distributions is outside the scope of this subsection. The alternative weight distribution explored is the normal (or Gaussian) distribution. The particular normal distribution used here is defined by a mean of 0 and a standard deviation of 1. To ensure a fair comparison with the uniform distribution previously used, the range of the weights allowed must be the same. To achieve this in the considered normal distribution, values higher than 1 and lower than -1 must be respectively bounded to 1 and -1. The resulting alternative weight distribution is applied to SLM-FLS and SLM-OLS, with and without sparseness.

Figure 6.12 presents the evolution plots with the training and generalization errors for SLM-FLS with a normal weight distribution, and with and without sparseness. To help the description the SLM versions that use the normal weight distribution are labeled as NWD (from normal weight distribution). The behavior of these variants is similar to the comparable variants that use an uniform weight distribution (shown in figure 6.6). No statistically significant differences are found when comparing the usage of these different weight distributions. Among the variants that use a normal weight distribution, the usage of sparseness has similar effects to the ones previously described. In terms of generalization, the only statistically significant difference is found in the PPB dataset, where using a high level of sparseness contributes to superior generalizations: 75% (p-value 4.846×10^{-4}), and 95% (p-value 6.161×10^{-5}). No significant influences are found in the Bio and LD50 datasets. Similarly to previous results, adding sparseness is found to be more influential in terms of training error. In the Bio dataset, sparseness levels of 25% or above achieve a training performance significantly superior than the variant without sparseness: 25% (p-value 2.552×10^{-9}), 50% (p-value 2.872×10^{-11}), 75% (p-value 2.872×10^{-11}), and 95% (p-value 2.872×10^{-11}). A similar result occurs in the PPB dataset when using sparseness levels of 50% (p-value 3.701×10^{-6}) and 75% (p-value 9.273×10^{-4}). No similar statistically significant differences are found in the LD50 dataset.

Similar conclusions are found when applying the normal weight distribution to SLM-OLS. Figure 6.13 presents the evolution plots with the training and generalization errors for SLM-OLS with a normal weight distribution, and with and without sparseness. As in the SLM-FLS results, the behavior of SLM-OLS variants is similar to the comparable variants that use an uniform weight distribution (shown in figure 6.10). No statistically significant differences are found when comparing the usage of these different weight distributions. The effect of using sparseness is consistent with what was previously seen in the SLM-OLS variants with uniform weight distributions. No statistically significant differences are found in terms of generalization. However, using any level of sparseness contributes to a better training data performance in the Bio dataset: 5% (p-value 2.065×10^{-4}), 25% (p-value 3.879×10^{-11}), 50% (p-value 2.872×10^{-11}), 75% (p-value 2.872×10^{-11}), and 95% (p-value 2.872×10^{-11}). In the PPB dataset, a similar effect is found when using sparseness levels of 50% or above: 50% (p-value 8.583×10^{-6}), 75% (p-value 1.732×10^{-4}), and 95% (p-value 3.131×10^{-7}). No statistically significant differences are found in the LD50 dataset.

6.3 Semantic Stopping Criteria

In iterative supervised learning algorithms it is common to reach a point in the search where no further induction seems to be possible with the available data. If the search is continued beyond this point, the risk of overfitting increases significantly. Even if the generalization simply stabilizes, no benefit exists in continuing the search. This section explores the usage of information extracted from the semantic neighborhood to determine when to stop the search. Each sample of the semantic neighborhood is gathered by repeatedly applying the semantic mutation operator. The size of the semantic neighborhood sample is the same as the SLM sample size parameter. Since SLM already creates a set of neighbors to be considered at each iteration, no further sampling is required. The equivalent case is also true for SSHC.

6.3.1 Stopping Criteria Definition

The first criterion is based on the variation of the error deviation of the models that are superior to the current best model. In this context, the term error deviation is used to refer to the standard deviation of the absolute errors of a given model over the training instances. Note that this criterion is only concerned with the models that improve over the current best model in a given iteration/generation. From these models, the crite-

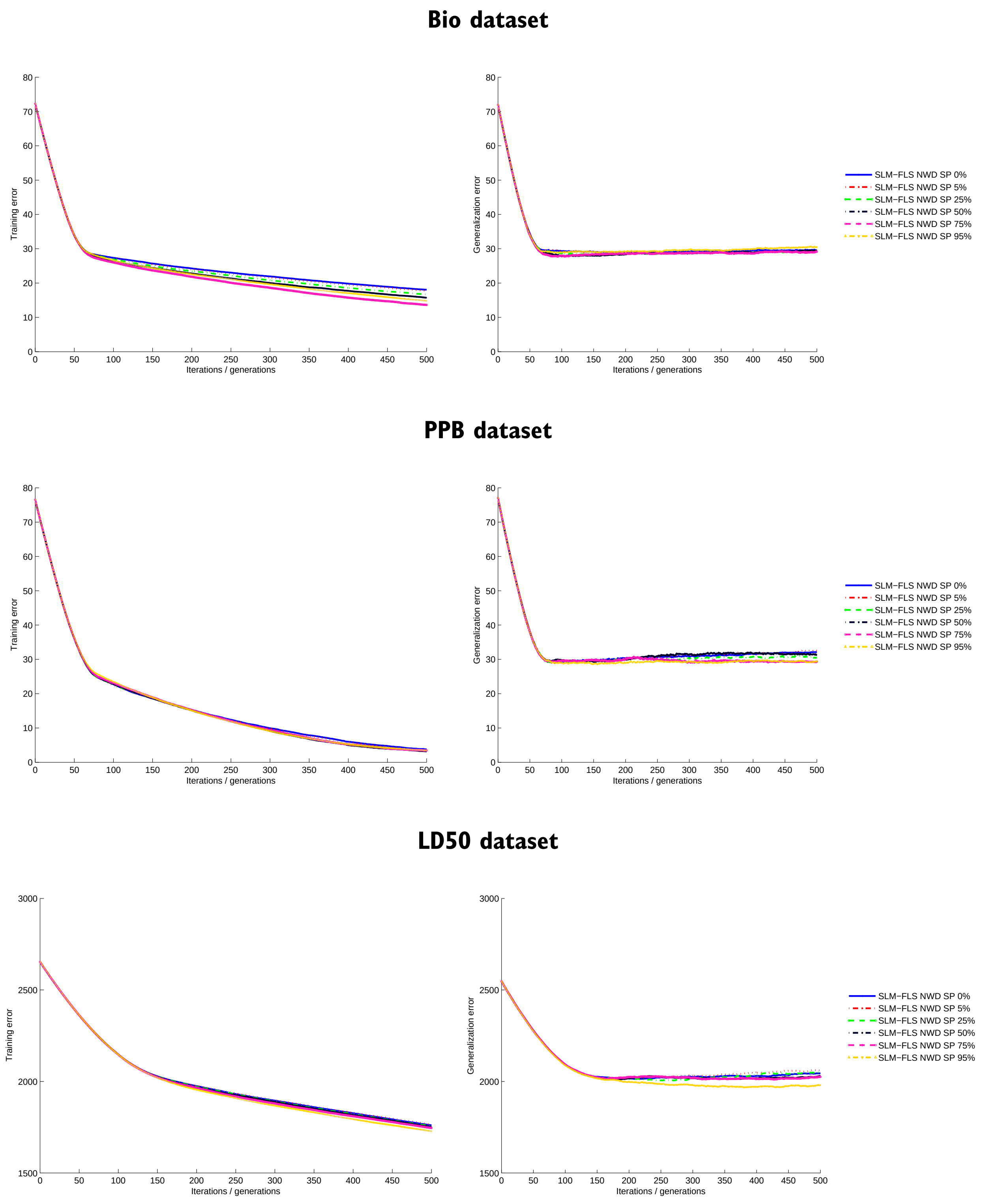


Figure 6.12: Training and generalization errors evolution plots for SLM-FLS NWD with and without sparseness

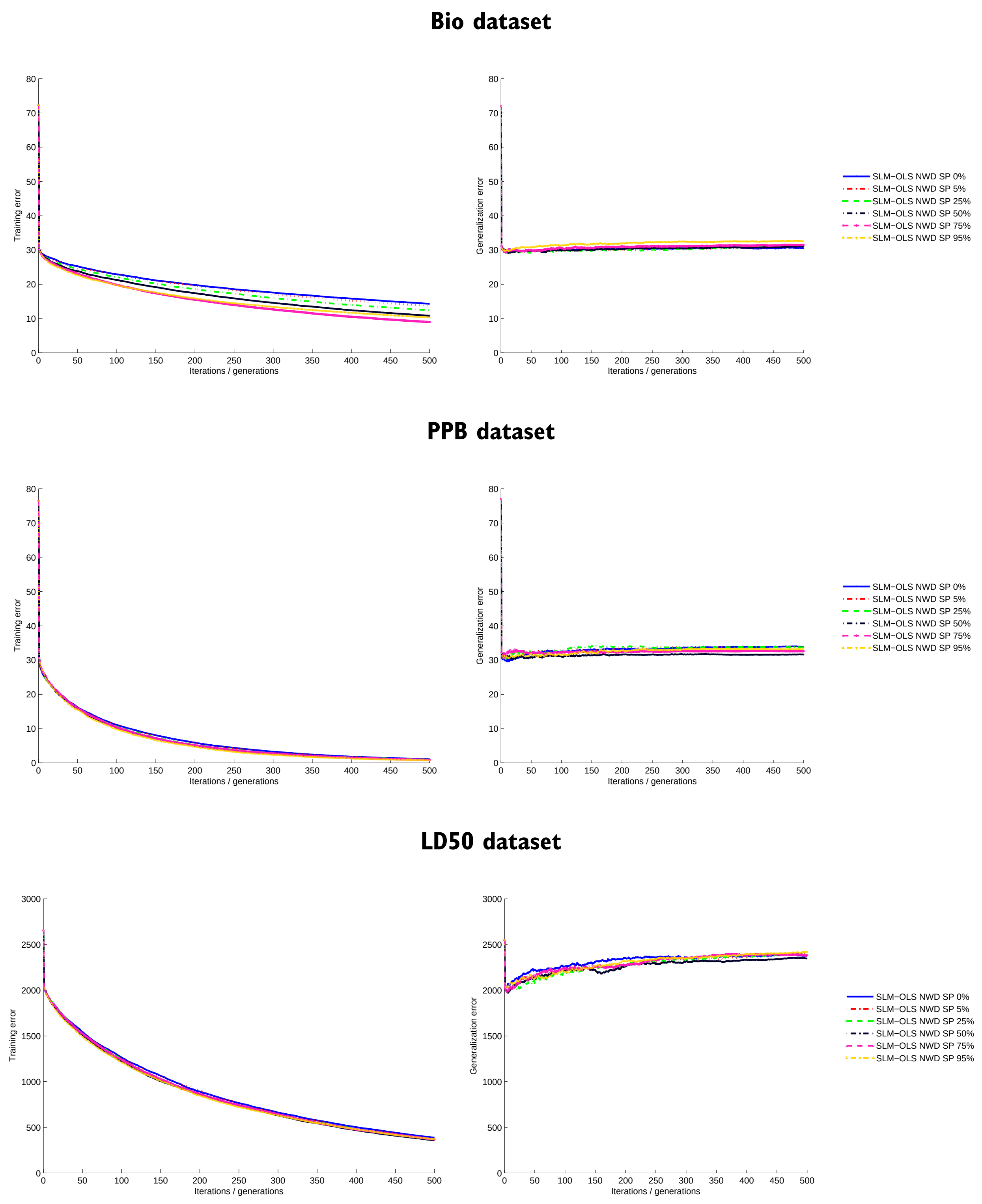


Figure 6.13: Training and generalization errors evolution plots for SLM-OLS NWD with and without sparseness

rion measures the percentage of models that reduce the error deviation in comparison with the error deviation of the current best model. This information allows to determine when the training error reduction starts to be conducted less uniformly across training instances. This may indicate that overfitting is starting to occur. The search is stopped when the criterion measure drops below a given threshold (parameter). This criterion is named Error Deviation Variation (EDV). Since this criterion is based on the error deviation, it does not prevent the algorithm from finding models with a large semantic (output) deviation, as this may actually be desired given the target semantics. The criterion also does not prevent the next best model to have a larger error deviation than the previous best, as this flexibility could be important in the learning process. The search is only stopped if a considerable majority of the models are improving the training error at the expense of larger error deviations.

The second criterion is based on measuring the effectiveness of the semantic variation operator used to perform the sampling. In this context, the effectiveness of the operator is defined as the percentage of times that the operator is able to produce a model that is superior to the current best model. During each iteration/generation, the effectiveness is measured with regard to the sample considered. The search is stopped when the effectiveness of the operator drops below a given threshold (parameter). This criterion is named Training Improvement Effectiveness (TIE). The reasoning underlying this criterion is that, if training error improvements are harder to find, then possibly these improvements are being forced at the expense of the resulting generalization.

6.3.2 Fixed Learning Step

This subsection studies the application of the proposed stopping criteria to SLM-FLS. Figure 6.14 presents the evolution of the measures used in the stopping criteria, and their complementary measures. These plots show the medians of each value throughout the runs if no stopping is applied. The left column presents the measures related with the EDV criterion, and the right column presents the measures related with the TIE criterion. The blue solid lines represent the measures that are directly used in the criteria to determine the stopping point. The red dashed lines are the respective complementary measures.

Starting with the EDV criterion (left column) in the Bio and PPB datasets, in the beginning of the runs the algorithm is very effective in generating models that are superior

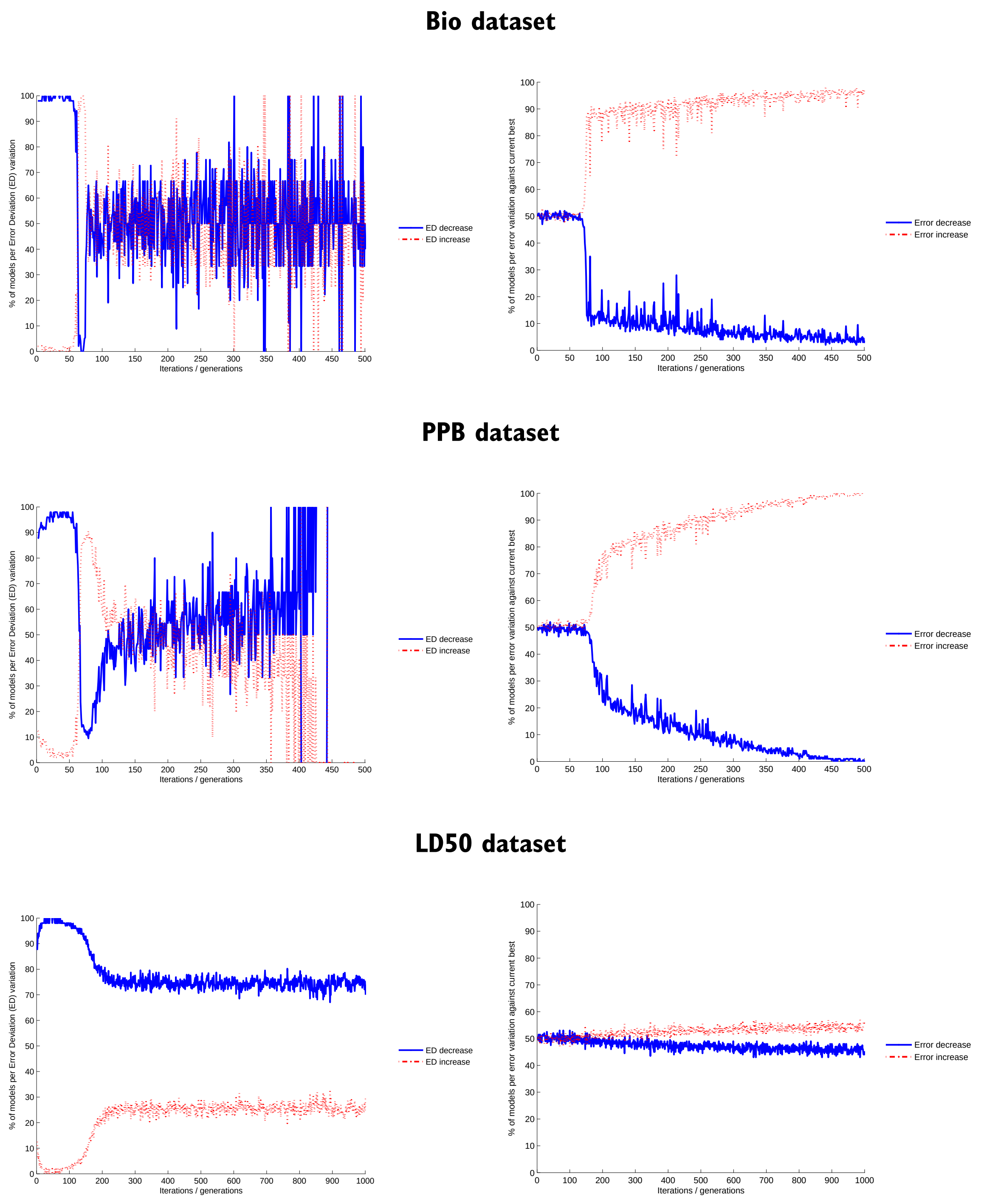


Figure 6.14: Evolution of the measures related to the EDV (left column) and the TIE (right column) criteria for SLM-FLS

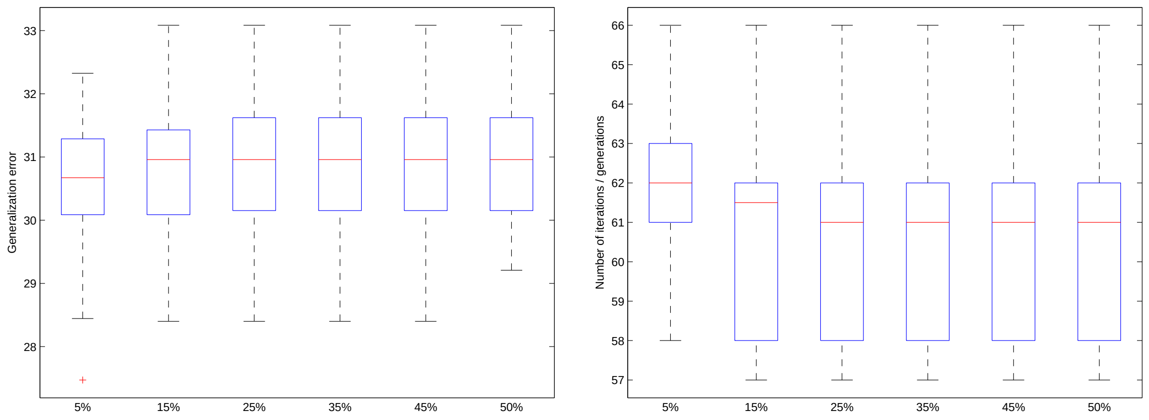
to the current best and, at the same time, reduce the error deviation (ED). This is shown by the line labeled as ED decrease. Until around iteration 50, the values of this measure are usually over 90% in both datasets. Between iterations 50 and 100, a quick decrease of this measure occurs, as well as the consequent increase of the complementary measure (labeled as ED increase). The ED decrease measure drops below 20%, while the ED increase measure increases to over 80%. This indicates that the algorithm is mostly finding models that are superior to the current best at the expense of increasing the error deviation. After this quick disruption, both measures seem to enter a phase where the values do not present a clear trend. Interestingly, this quick disruption coincides with the generalization plateau, where the generalization error stabilizes while the training error keeps decreasing (as shown in figure 6.8). This might be an interesting stopping point as no further induction seems to be possible. Continuing the search beyond this point might only increase the model size and the computational time. In the LD50 dataset, the ED decrease measure also starts at very high values. However, a similar quick disruption is not apparent from the median values presented. It will be clear ahead that a quick disruption also occurs in the LD50 dataset. The fact that this disruption happens at considerably different iterations in different runs, makes the median values misleading.

In the TIE criterion (right column), a clear pattern also occurs in the Bio and PPB datasets. Initially, the effectiveness of the variation operator (labeled as error decrease) is around 50%. In other words, generating a model which is superior to the current best is approximately as likely as generating a model which is inferior to the current best. This should be expected in the fixed learning step version of the GSM-NN operator, as in mutation operators with this characteristic approximately 50% of the models are generated in the direction of the target semantics, and the other 50% are generated in the opposite direction. Similarly to what occurs in the EDV criterion, a disruption of this scenario happens between iterations 50 and 100. The main difference is that the disruption in the TIE criterion occurs a few iterations later than in the EDV criterion. As in the EDV criterion, this disruption coincides with the generalization plateau. After this disruption the effectiveness of the variation operator drops below 20% in the Bio dataset, and below 30% in the PPB dataset. In the LD50 the effectiveness of the variation operator also starts around 50%. This measure drops slightly during the run, but no quick disruption of the median values occurs as in the other datasets. Similarly to what happens in the EDV criterion, the apparent lack of disruption is related to the considerable different behaviors in different runs. It will be clear ahead that quick disruptions also occur in the LD50 dataset.

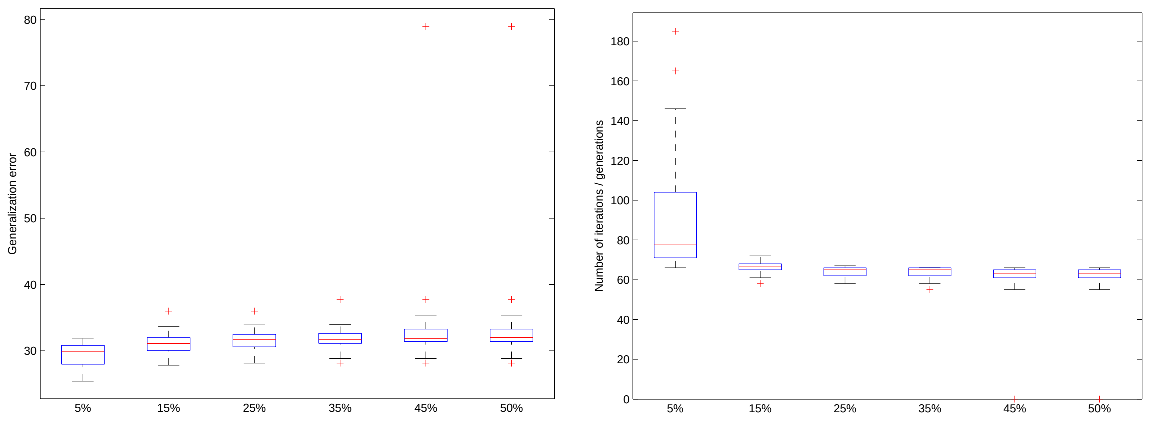
Figure 6.15 presents the boxplots with the generalization errors achieved by applying the EDV criterion with different stopping thresholds. The same figure presents the number of iterations conducted until the stopping occurred. The values tested for the stopping thresholds are: 5%, 15%, 25%, 35%, 45%, and 50%. The exploration of different values for the stopping threshold is aimed at assessing the robustness of each criterion in regard to this parameter. Across all datasets, stopping always occurs with competitive generalization values for reasonable stopping thresholds. It should be expected that extreme stopping thresholds (5%, 45%, and 50%) could lead to less robust outcomes. Because of this, middle values (such as 25%) are preferred. For the remainder of this chapter, a stopping threshold of 25% is used by default. Also across all datasets, the 25% stopping threshold and the stopping thresholds around it (15% and 35%) achieve similar outcomes. This is a desirable outcome as no tuning of the stopping threshold seems to be required. In the Bio dataset, stopping occurs between iterations 57 and 66 for every stopping threshold. All stopping thresholds achieve median generalizations of around 31 RMSE. In the PPB dataset, stopping occurs around iteration 60 for every stopping threshold except for 5% which presents a larger stopping variation. All stopping thresholds achieve median generalizations of around 32 RMSE. Since stopping takes longer in the LD50 dataset, the runs were extended to 1000 iterations. As previously mentioned, there is more variation among different runs in the LD50 dataset. Stopping occurs between iterations 200 and 1000. With a 5% stopping threshold, some runs do not terminate because of the stopping criterion, but rather because of the iterations limit imposed. All stopping thresholds achieve median generalizations of around 2000 RMSE.

Figure 6.16 presents the boxplots with the results achieved by applying the TIE criterion with different stopping thresholds. As in the EDV criterion, stopping always occurs with competitive generalization values for reasonable stopping thresholds. This is valid across all datasets. As a general trend, using the TIE criterion results in later stopping points than in the EDV criterion. Considering the default stopping threshold of 25%, in the Bio dataset stopping occurs around iteration 72. The median generalization achieved is around 30 RMSE. In the PPB dataset, stopping occurs around iteration 85. The median generalization achieved is around 29 RMSE. As in the EDV criterion, the stopping points in the LD50 dataset present a bigger variation among different runs. Stopping oc-

Bio dataset



PPB dataset



LD50 dataset

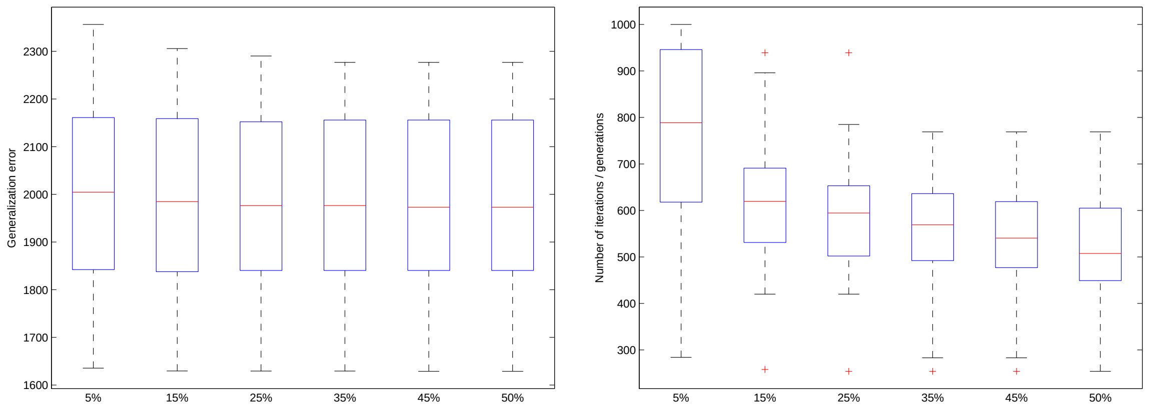


Figure 6.15: Boxplots with the generalization errors (left column) and the number of iterations (right column) resulting from the application of the EDV criterion with different stopping thresholds to SLM-FLS

curs around iteration 600 for the default stopping threshold. The median generalization achieved is around 2000 RMSE. It is clear in the application of the TIE criterion that using extreme stopping thresholds (5%, 45%, and 50%) leads to less robust outcomes across all datasets.

Figure 6.17 presents a comparison of the results achieved by applying both stopping criteria with the default stopping threshold. The TIE criterion achieves statistically significant superiority in terms of generalization in the Bio and PPB datasets (p-values: Bio $\mathbf{3.878 \times 10^{-4}}$, and PPB $\mathbf{9.899 \times 10^{-7}}$). This is achieved by allowing the search to run for more iterations. The differences in the number of iterations conducted are also statistically significant (p-values: Bio $\mathbf{2.219 \times 10^{-11}}$, and PPB $\mathbf{2.384 \times 10^{-11}}$). In the LD50 dataset, there are no statistically significant differences in terms of generalization and number of iterations. Since both criteria stop at similar iterations, it should be expected that they achieve similar generalizations.

Overall, both stopping criteria are effective at detecting stopping points that result in a competitive generalization. Both stopping criteria perform well when applied to SLM-FLS. In a comparison between both criteria, the TIE criterion generalizes better in 2 out of the 3 datasets.

6.3.3 Optimal Learning Step

This subsection studies the application of the proposed stopping criteria to SLM-OLS. Figure 6.18 presents the evolution of the measures used in the stopping criteria, and their complementary measures.

In the EDV criterion (left column), the percentage of models that reduce the error deviation starts at high values. This is similar to what happens in the same criterion for SLM-FLS. In SLM-OLS, a quick disruption of the error deviation decrease measure also occurs. This is particular clear in the Bio and LD50 datasets. In the Bio dataset, this disruption occurs with less than 10 iterations conducted. In the LD50 dataset, the disruption occurs after less than 20 iterations. In the PPB dataset, the disruption does not appear to be as clear and it only seems to occur around iteration 80. However, it will be clear ahead that quick disruptions occur across all datasets, and that the error deviation decrease measure is effective at detecting stopping points that result in a competitive generalization. To improve the readability of the measures associated with the EDV criterion,

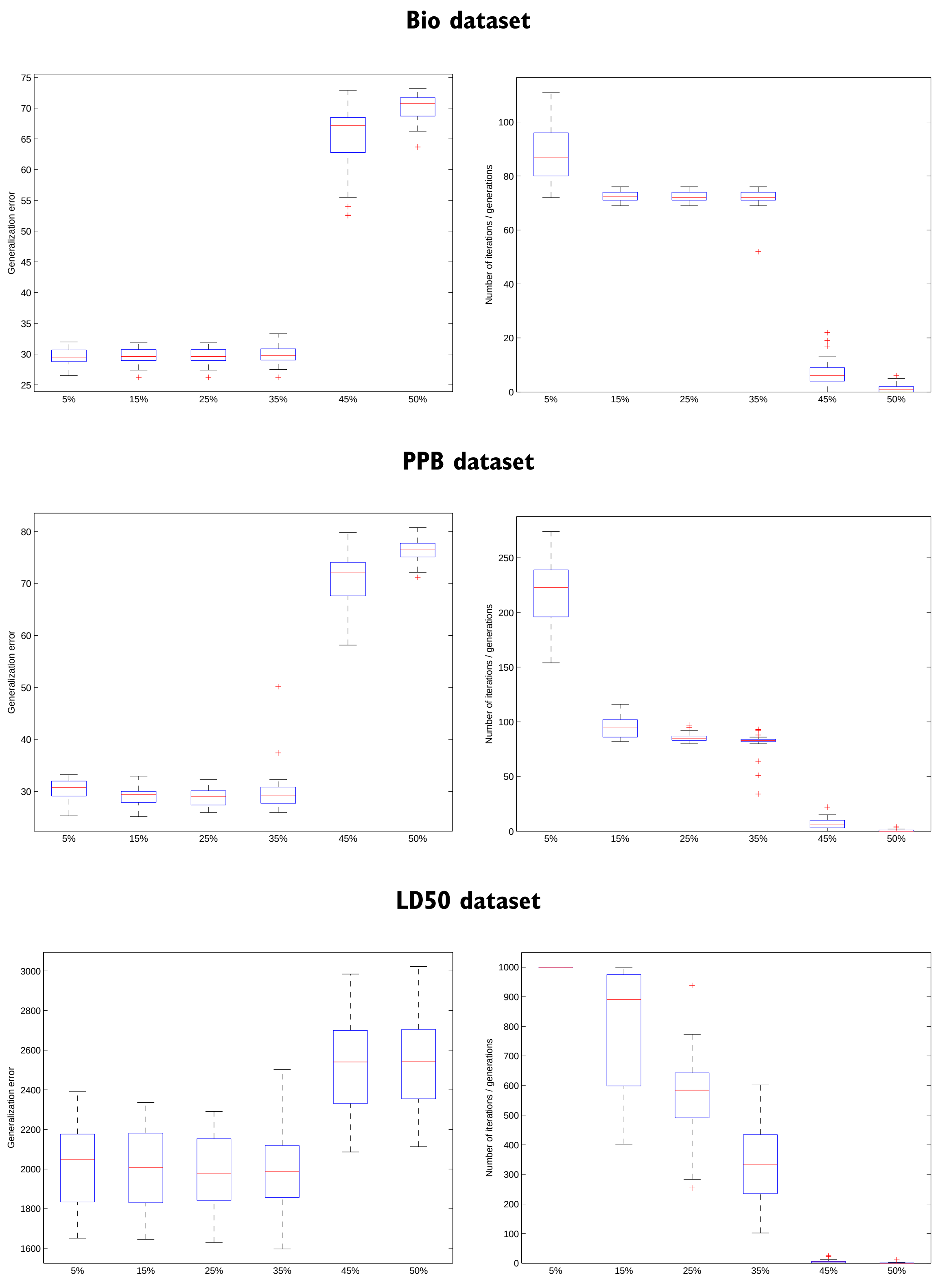


Figure 6.16: Boxplots with the generalization errors (left column) and the number of iterations (right column) resulting from the application of the TIE criterion with different stopping thresholds to SLM-FLS

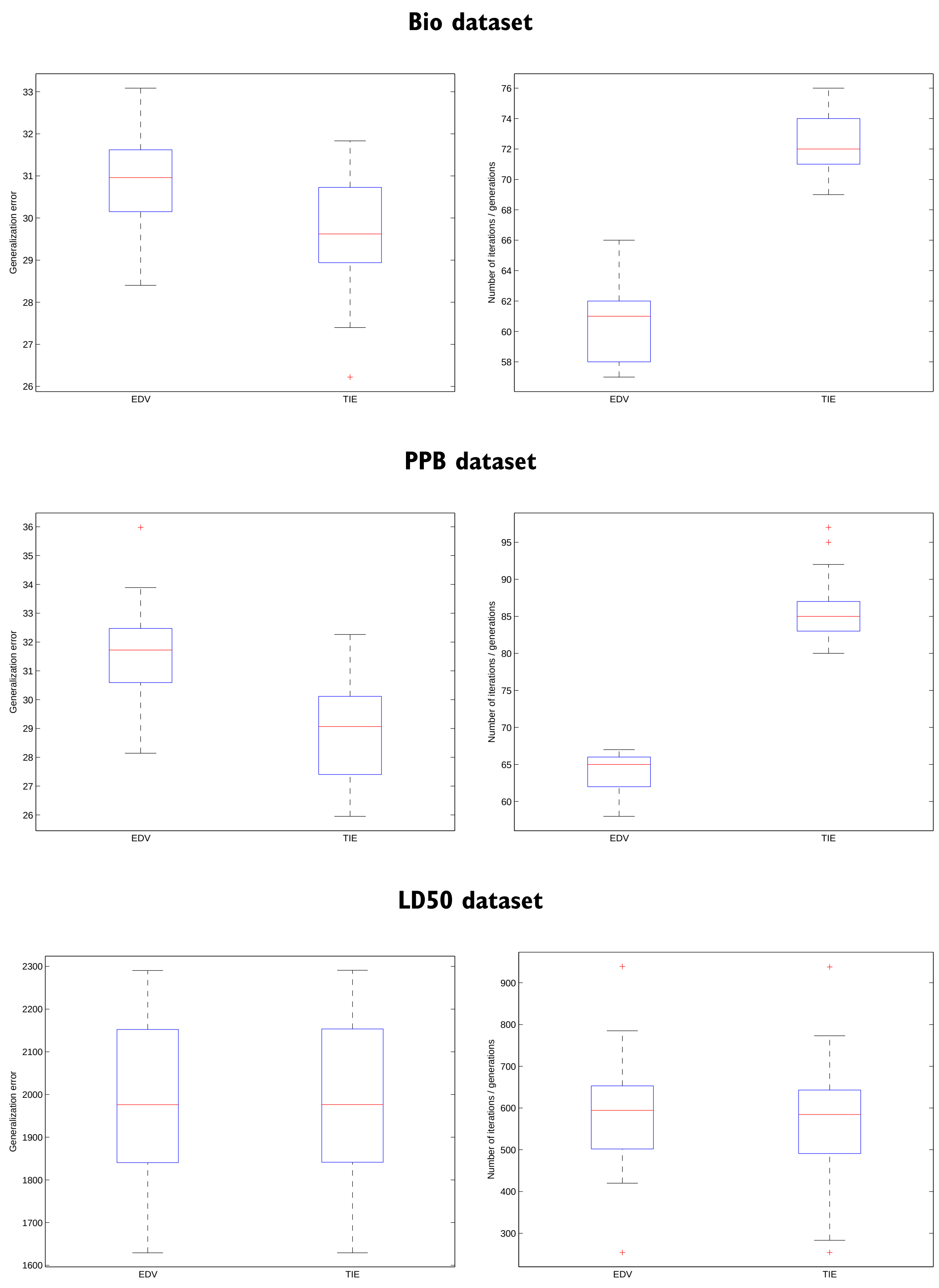


Figure 6.17: Boxplots with the generalization errors (left column) and the number of iterations (right column) resulting from the application of both stopping criteria with a 25% stopping threshold to SLM-FLS

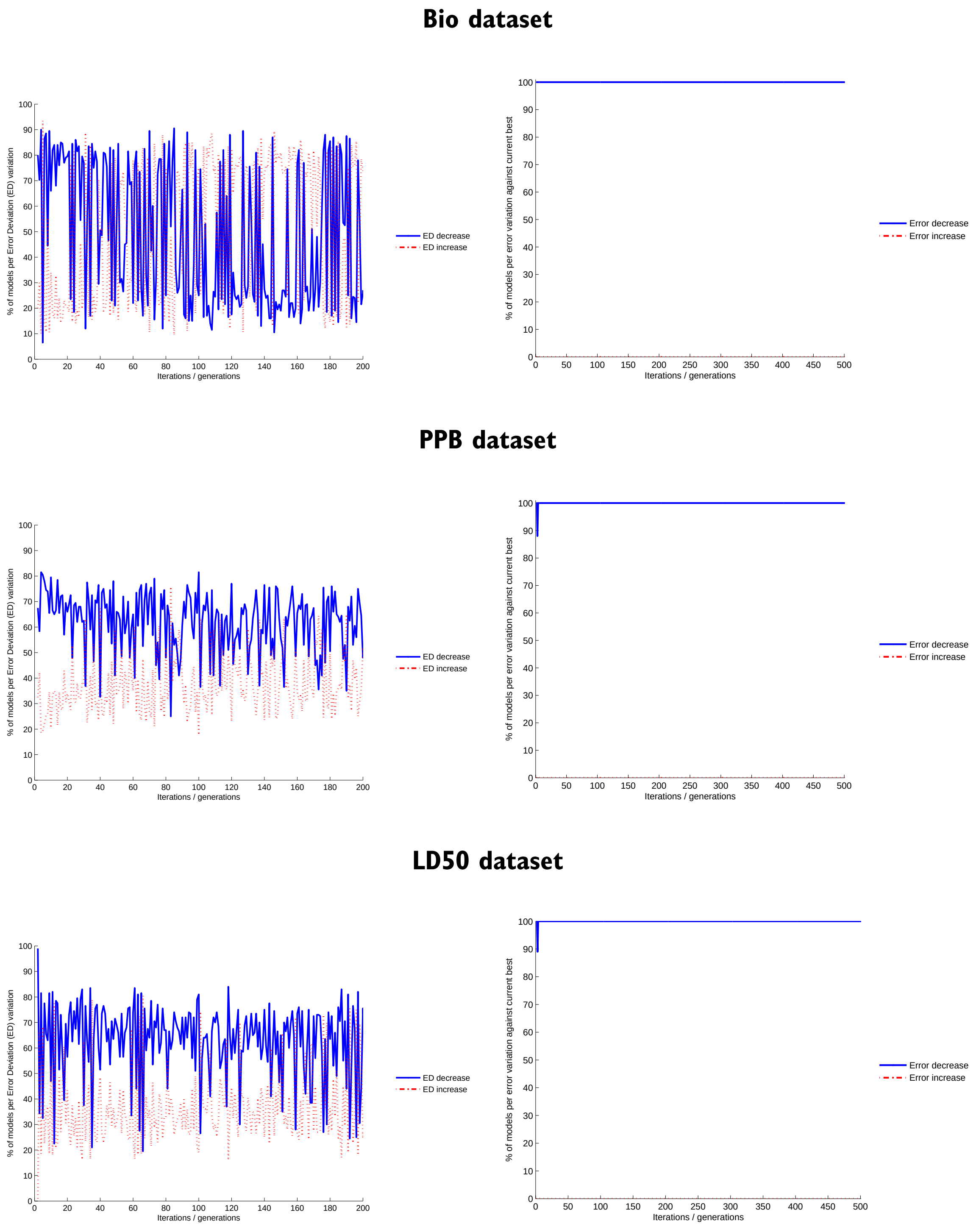


Figure 6.18: Evolution of the measures related to the EDV (left column) and the TIE (right column) criteria for SLM-OLS

the plots in the left column only show the first 200 iterations. This allows for a better understanding of where the initial disruptions occur. The behavior of the measures after iteration 200 is similar to the behavior between iterations 100 and 200.

The effectiveness of the variation operator used in the TIE criterion (right column), is almost always 100%. Similar effectiveness were presented in the previous chapter for the equivalent optimal mutation step operators. Because of this very high effectiveness, the TIE criterion is not able to provide interesting stopping points under reasonable stopping thresholds. Therefore, the TIE criterion is not appropriate to use in methods that apply similar optimal computation of learning/mutation steps.

Figure 6.19 presents the boxplots with the results achieved by applying the EDV criterion with different stopping thresholds. Across all datasets, stopping always occurs with competitive generalization values for reasonable stopping thresholds. As presented in the SLM-FLS case, the stopping thresholds around the default threshold (15% and 35%) achieve similar outcomes to the default threshold. The extreme stopping threshold of 5%, presents a considerable bigger variation in the stopping points. In the PPB and LD50 datasets, using the 5% stopping threshold results in some runs not stopping before the iterations limit imposed. Considering the default stopping threshold, in the Bio dataset stopping occurs around iteration 3. The median generalization achieved is around 30 RMSE. In the PPB dataset, stopping occurs around iteration 5. The median generalization achieved is around 32 RMSE. In the LD50 dataset, stopping occurs around iteration 3. The median generalization achieved is around 2000 RMSE. As in the SLM-FLS case, the EDV criterion is also effective when applied to SLM-OLS. It is able to detect stopping points that result in a competitive generalization.

Figure 6.20 presents a comparison of the results achieved by applying both stopping criteria to SLM-FLS and SLM-OLS. The results presented are based on the usage of the default stopping threshold. To help the description the following simplifications are used: SLM-FLS EDV is used to describe SLM-FLS using the EDV criterion; SLM-FLS TIE is used to describe SLM-FLS using the TIE criterion; SLM-OLS EDV is used to describe SLM-OLS using the EDV criterion. In the Bio dataset, no statistically significant differences are found in terms of generalization in the comparison between SLM-OLS EDV and both criteria applied to SLM-FLS. In the PPB dataset, SLM-FLS TIE achieves a superior generalization in comparison with SLM-OLS EDV (p-value 3.274×10^{-4}). No statistically significant differences exist in the comparison between SLM-FLS EDV and SLM-OLS EDV. In the

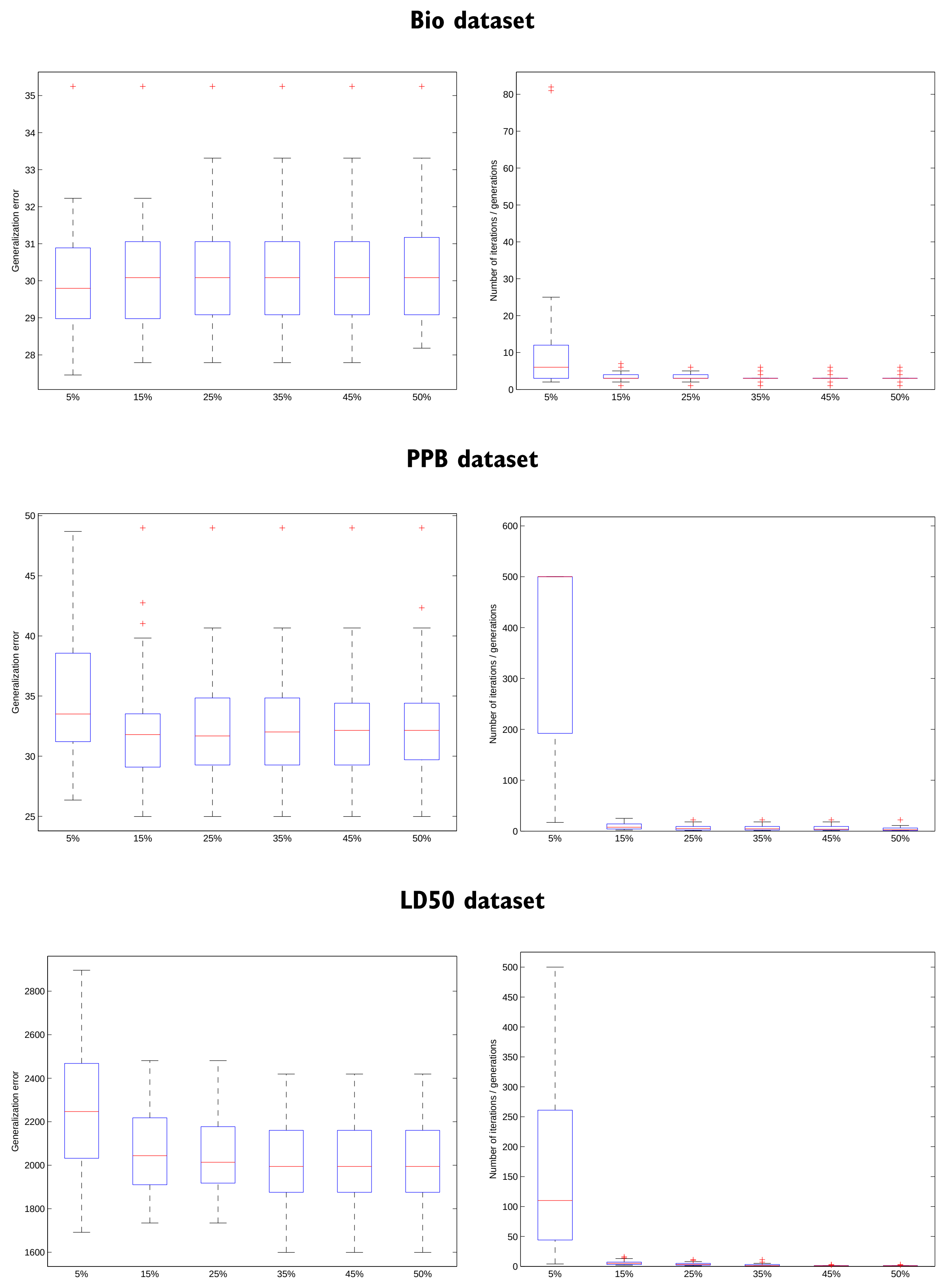


Figure 6.19: Boxplots with the generalization errors (left column) and the number of iterations (right column) resulting from the application of the EDV criterion with different stopping thresholds to SLM-OLS

LD50 dataset, no statistically significant differences are found in terms of generalization in the comparison between SLM-OLS EDV and both criteria applied to SLM-FLS. Across all datasets, SLM-OLS EDV stops significantly faster than SLM-FLS EDV (p-values: Bio $\mathbf{1.269 \times 10^{-11}}$, PPB $\mathbf{2.367 \times 10^{-11}}$, and LD50 $\mathbf{2.557 \times 10^{-11}}$) and SLM-FLS TIE ($p$-values: Bio $\mathbf{1.389 \times 10^{-11}}$, PPB $\mathbf{2.611 \times 10^{-11}}$, and LD50 $\mathbf{2.557 \times 10^{-11}}$).

Overall, the application of the EDV criterion to SLM-OLS is effective at detecting stopping points that result in a competitive generalization. This criterion application results in significantly faster stopping points than in the SLM-FLS variants. In 2 out of the 3 datasets, these faster stopping points do not compromise the generalization achieved. The application of the TIE criterion to SLM-FLS in the PPB dataset, is the only case where a different variant can achieve a significantly superior generalization.

6.3.4 Stopping in Geometric Semantic Genetic Programming

This subsection explores the application of the proposed stopping criteria to the versions of GSGP equivalent to SLM-FLS (SSHC BM) and SLM-OLS (SSHC ABM). The evolution of the measures used in the stopping criteria are presented in figure 6.21 for SSHC BM, and in figure 6.22 for SSHC ABM.

Starting with SSHC BM, the evolution of the measures associated with both criteria presents similarities with what happens in SLM-FLS. In the EDV criterion (left column), the percentage of models that reduce the error deviation starts at high values across all datasets. A quick decrease of this measure occurs in the beginning of the runs. This decrease is particularly quick in the Bio and PPB datasets. As in SLM-FLS, the decrease in the LD50 dataset is less sharp. The behavior of the measures associated with the TIE criterion (right column) is also similar to what happens in SLM-FLS. The effectiveness of the variation operator (labeled as error decrease) starts around 50%. This effectiveness then drops consistently throughout the runs. The decrease of the effectiveness of the variation operator is particularly clear in the Bio and PPB datasets. In the LD50 dataset, this same decrease is slower but consistent. In SSHC ABM, the evolution of the measures associated with both criteria is very similar to what occurs in SLM-OLS. In the EDV criterion, similar disruptions occur across all datasets. Regarding the TIE criterion, the effectiveness of ABM is always very close to 100%, as seen in the previous chapter. As in SLM-OLS, the TIE criterion is not appropriate to use in SSHC ABM given the very high

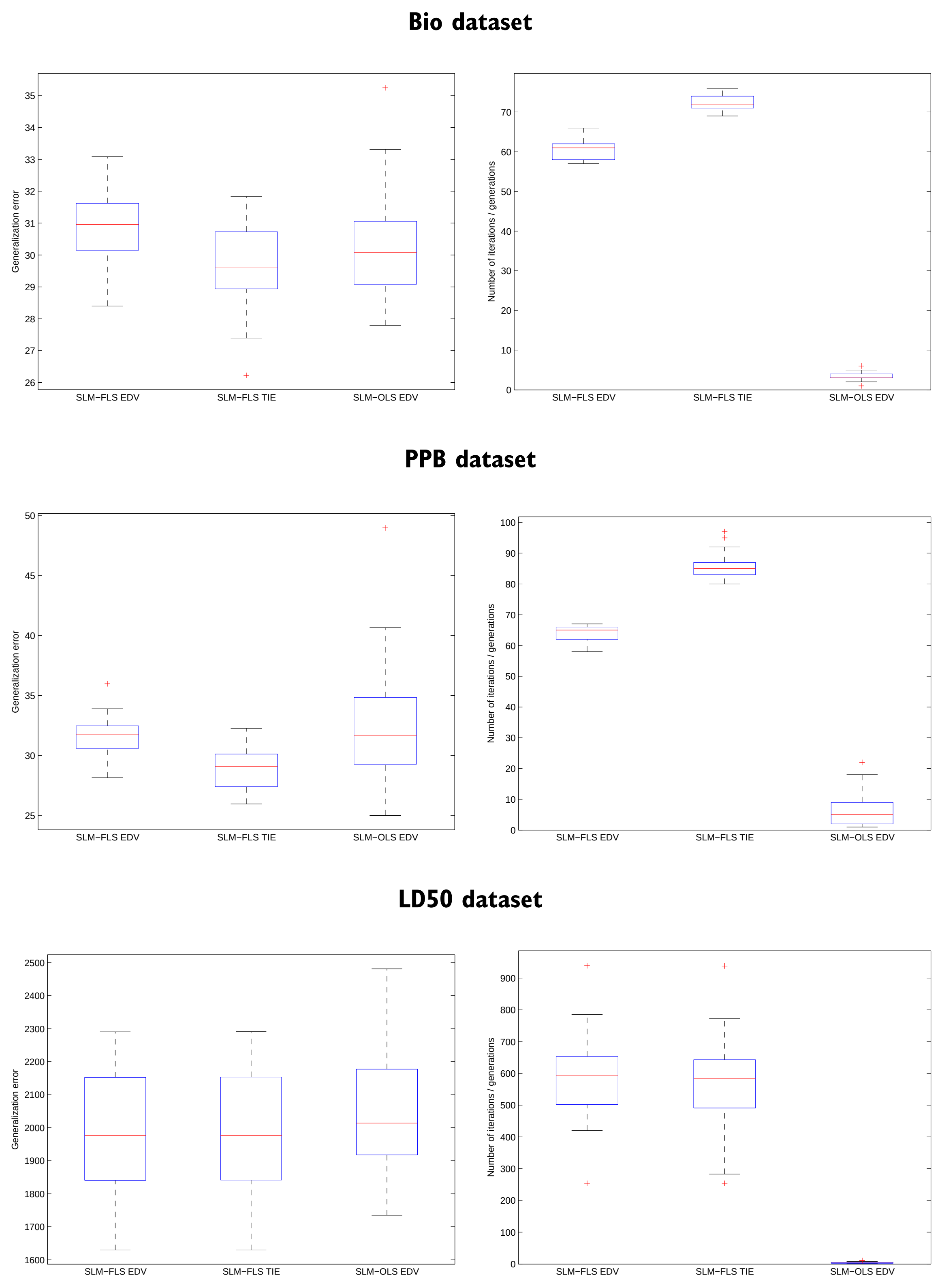
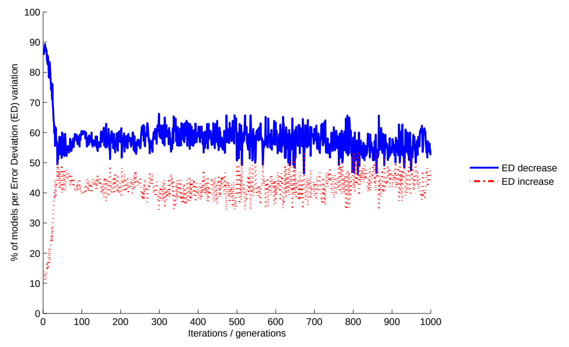
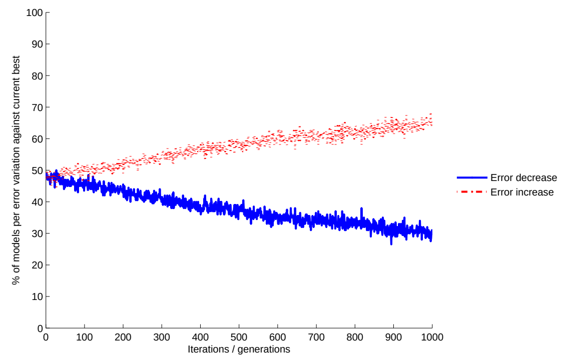


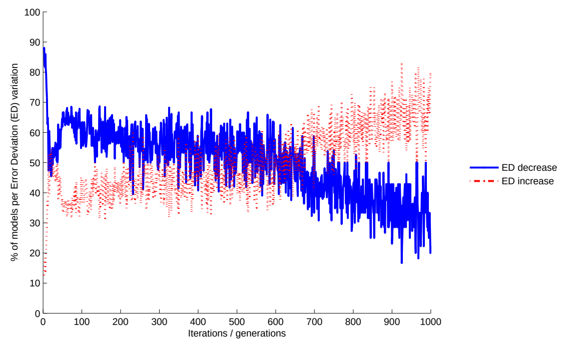
Figure 6.20: Boxplots with the generalization errors (left column) and the number of iterations (right column) resulting from the application of both stopping criteria with a 25% stopping threshold to SLM-FLS and SLM-OLS

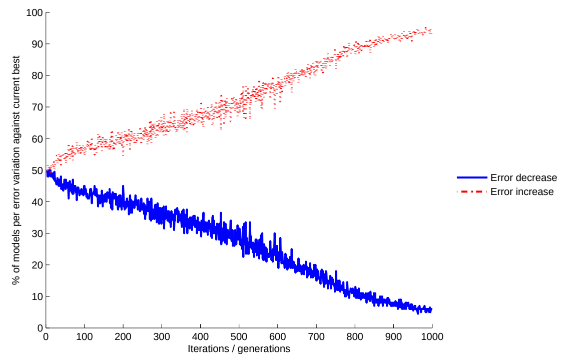
Bio dataset



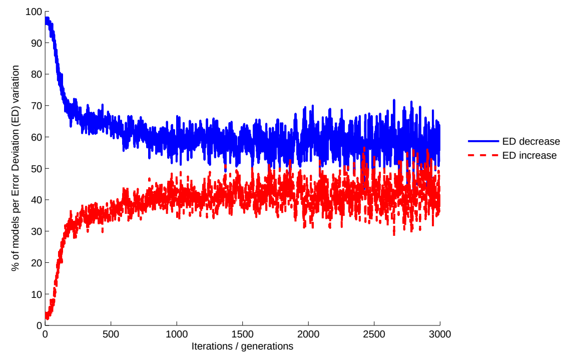


PPB dataset





LD50 dataset



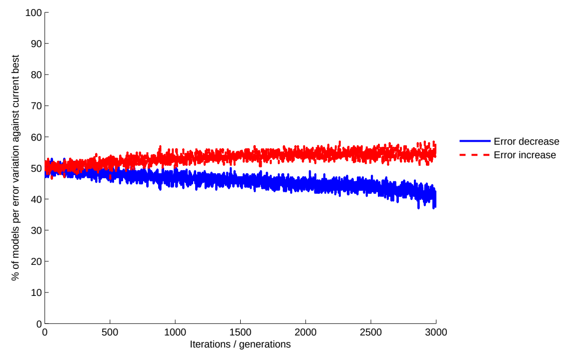


Figure 6.21: Evolution of the measures related to the EDV (left column) and the TIE (right column) criteria for SSHC BM

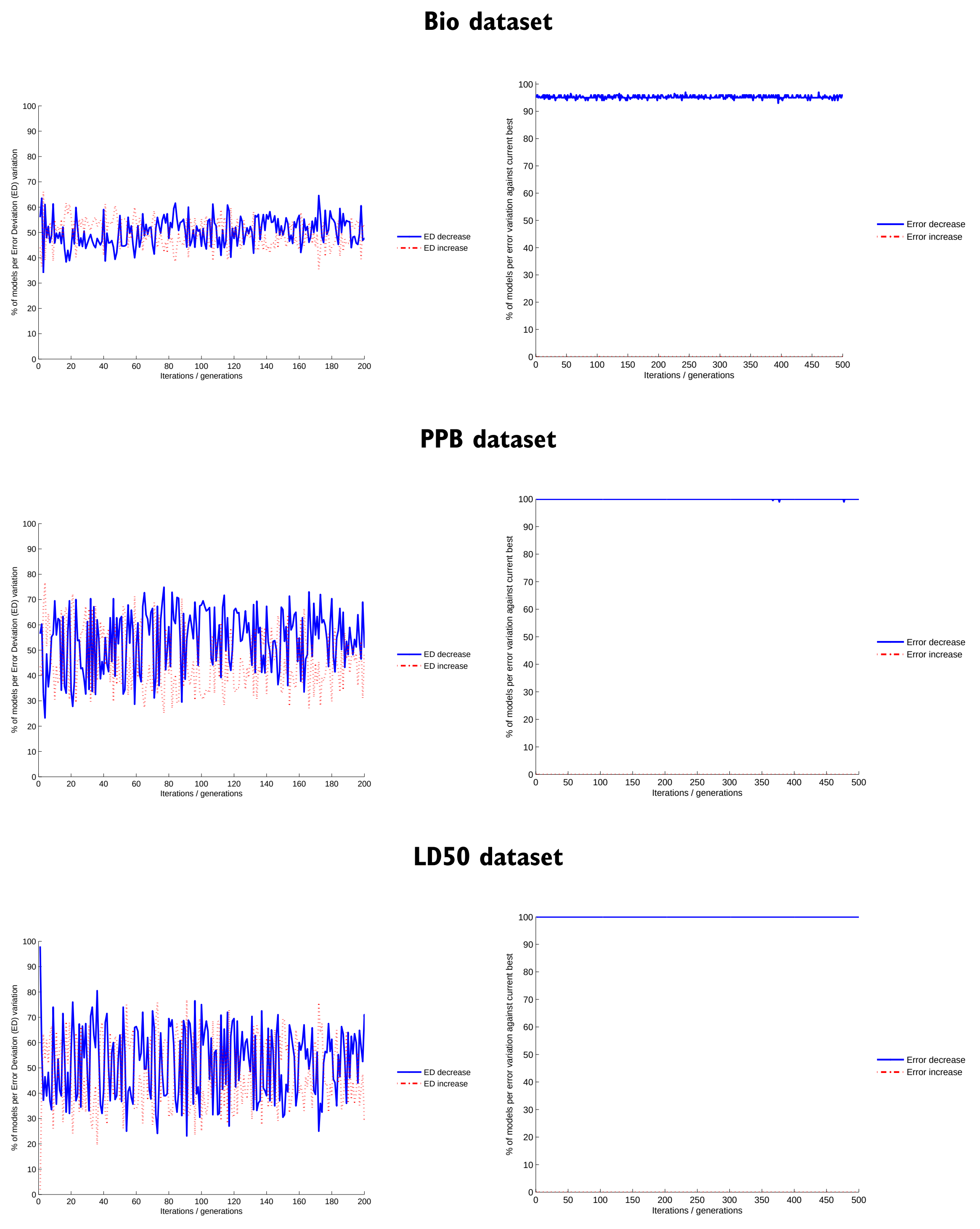


Figure 6.22: Evolution of the measures related to the EDV (left column) and the TIE (right column) criteria for SSHC ABM

effectiveness of the associated variation operator.

Figure 6.23 presents a comparison of the results achieved by applying both stopping criteria to SSHC BM and SSHC ABM. The results presented are based on the usage of the default stopping threshold. To help the description the following simplifications are used: SSHC BM EDV is used to describe SSHC BM using the EDV criterion; SSHC BM TIE is used to describe SSHC BM using the TIE criterion; SSHC ABM EDV is used to describe SSHC ABM using the EDV criterion. In the Bio and PPB datasets, SSHC BM TIE achieves significantly superior generalizations than SSHC BM EDV (p-values: Bio $\mathbf{5.445 \times 10^{-3}}$, and PPB $\mathbf{3.585 \times 10^{-3}}$) and SSHC ABM EDV ($p$-values: Bio $\mathbf{7.476 \times 10^{-6}}$, and PPB $\mathbf{6.728 \times 10^{-4}}$). In the same datasets, no statistically significant differences are found in the comparison between SSHC BM EDV and SSHC ABM EDV. In the LD50 dataset, SSHC BM EDV generalizes better than SSHC BM TIE (p-value $\mathbf{4.969 \times 10^{-3}}$). No statistically significant differences are found in the comparisons between SSHC BM EDV and SSHC ABM EDV, and between SSHC ABM EDV and SSHC BM TIE. Across all datasets, SSHC ABM EDV stops significantly faster than SSHC BM EDV (p-values: Bio $\mathbf{4.272 \times 10^{-8}}$, PPB $\mathbf{3.442 \times 10^{-10}}$, and LD50 $\mathbf{6.544 \times 10^{-10}}$) and SSHC BM TIE ($p$-values: Bio $\mathbf{2.836 \times 10^{-11}}$, PPB $\mathbf{2.638 \times 10^{-11}}$, and LD50 $\mathbf{2.243 \times 10^{-11}}$). In the PPB and LD50 datasets, SSHC BM EDV stops significantly faster than SSHC BM TIE (p-values: PPB $\mathbf{2.325 \times 10^{-9}}$, and LD50 $\mathbf{3.879 \times 10^{-11}}$). No statistically significant differences are found in the same comparison in the Bio dataset. Overall, the stopping criteria are also effective in detecting interesting stopping points in GSGP as the search is always stopped before overfitting occurs. Across all datasets, the best SSHC variant always achieves a competitive generalization.

A final note on the direct comparisons between equivalent SLM and SSHC variants when using the same stopping criterion. SLM-FLS EDV generalizes better than SSHC BM EDV in the Bio (p-value $\mathbf{2.211 \times 10^{-3}}$) and PPB ($p$-value $\mathbf{5.715 \times 10^{-4}}$) datasets. SLM-FLS EDV stops faster in the LD50 dataset (p-value $\mathbf{9.845 \times 10^{-6}}$). No statistically significant differences are found in the remaining comparisons between these methods. SLM-FLS TIE generalizes better than SSHC BM TIE in the PPB (p-value $\mathbf{2.759 \times 10^{-4}}$) and LD50 ($p$-value $\mathbf{3.710 \times 10^{-5}}$) datasets. SLM-FLS TIE stops faster across all datasets (p-values: Bio $\mathbf{2.628 \times 10^{-11}}$, PPB $\mathbf{5.006 \times 10^{-11}}$, and LD50 $\mathbf{2.872 \times 10^{-11}}$). No statistically significant differences are found in the remaining comparisons between these methods. SLM-OLS EDV generalizes better than SSHC ABM EDV in the Bio (p-value $\mathbf{1.256 \times 10^{-8}}$) and PPB ($p$-value $\mathbf{1.723 \times 10^{-3}}$) datasets. SLM-OLS EDV stops faster in the Bio dataset

(p-value 1.292×10^{-5}), while SSHC ABM EDV stops faster in the PPB dataset (p-value 1.376×10^{-4}). No statistically significant differences are found in the remaining comparisons between these methods. Overall, the SLM variants are more robust than the equivalent GSGP variants.

6.4 Extended Comparison

This section presents a comparison between the SLM variants and some non-evolutionary supervised learning methods. The non-evolutionary methods tested are: Support Vector Regression (SVR); Multilayer Perceptron (MLP) using backpropagation; Random Tree (RT); and Linear Regression (LR). These methods are tested using their respective WEKA (Hall et al., 2009) implementations. The default parameters are used. The SVR version tested uses the polynomial kernel with the exponent parameter set to 1. The complexity parameter (C) is also set to 1. The MLP version tested uses the following parameters: learning rate of 0.3; momentum rate of 0.2; 500 epochs (iterations); and the number of neurons in the hidden layer is approximately set to half of the number of attributes in each dataset. The RT method implemented in WEKA constructs random trees by randomly choosing attributes at each tree node. The SLM variants tested use the stopping criteria with the default stopping threshold. To help the description the following simplifications are used: FLS EDV is used to describe SLM-FLS using the EDV criterion; FLS TIE is used to describe SLM-FLS using the TIE criterion; OLS EDV is used to describe SLM-OLS using the EDV criterion. The generalization is assessed by performing a 30-fold cross-validation. Although using 10 folds is more common, performing at least 30 runs is important to assess the behavior of stochastic methods such as the SLM. Because of this, all methods run 30 times, one for each fold. Note that the SLM generalization errors presented might be slightly better than in the previous sections because the amount of training data provided is now higher (70% before against over 96% now).

Table 6.2 presents the median and average generalization errors achieved by all methods across the 30 folds. Across all datasets, the SLM variants achieve the lowest average generalization errors. In terms of median generalization error, a similar outcome is achieved. The only case where a SLM variant is not superior to all other methods occurs in the LD50 dataset, where FLS TIE has a slightly higher median than MLP (1824.0 against 1803.5). Across all datasets, the best SLM variant always achieves the best average and

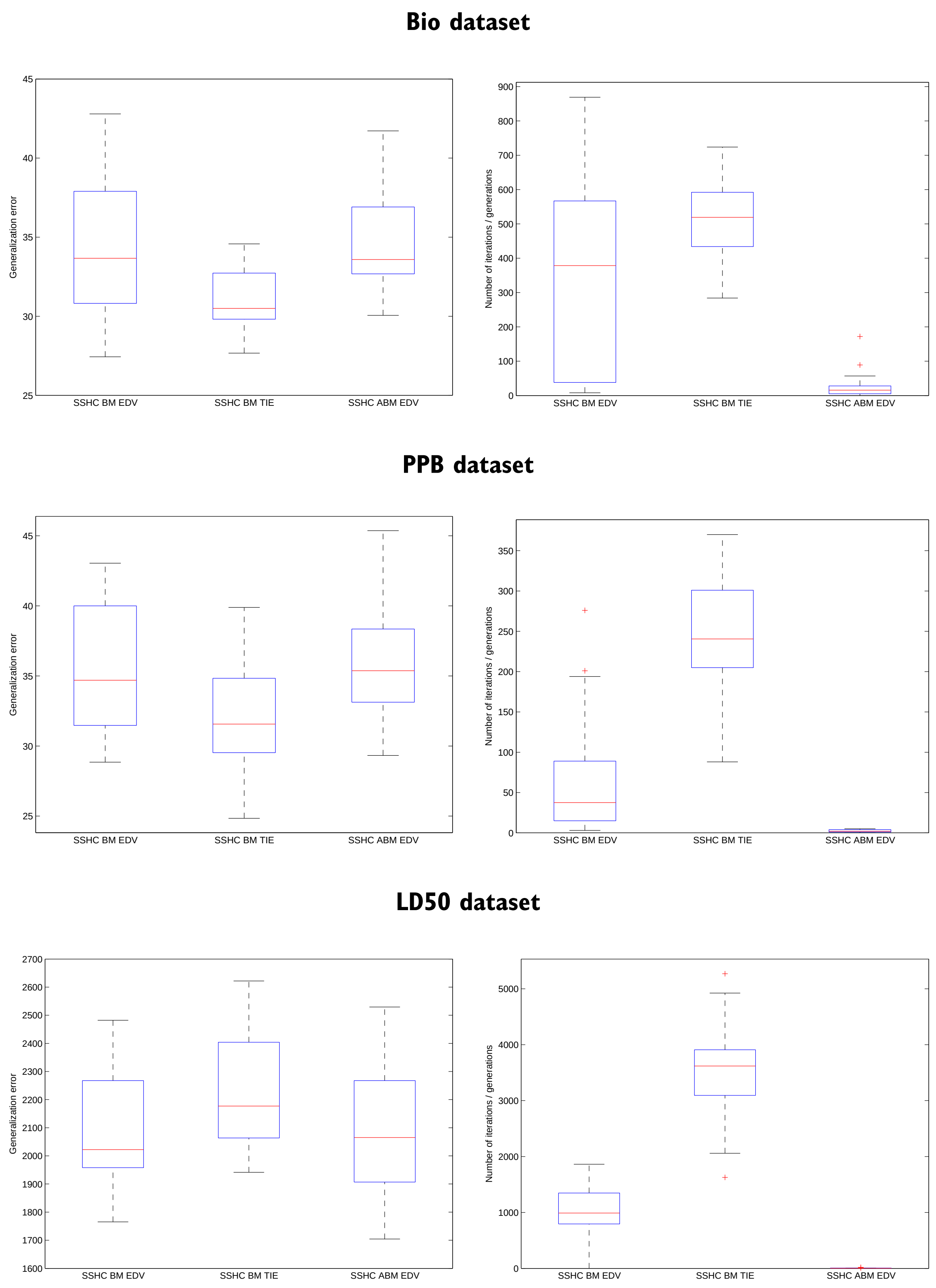


Figure 6.23: Boxplots with the generalization errors (left column) and the number of generations (right column) resulting from the application of both stopping criteria with a 25% stopping threshold to SSHC BM and SSHC ABM

median generalization errors of all the methods.

Table 6.2: Median and average generalization errors across the 30 folds

Dataset	**Measure**	**FLS EDV**	**FLS TIE**	**OLS EDV**	**SVR**	**MLP**	**RT**	**LR**
Bio	Median	30.6	28.9	28.9	34.8	37.2	35.7	34.0
	Average	30.6	29.2	30.5	39.0	1375.2	36.8	42.9
PPB	Median	30.3	26.0	29.8	33.1	36.4	39.6	39.0
	Average	30.8	27.0	30.5	1029.5	35.8	37.7	3615.3
LD50	Median	1796.7	1824.0	1782.6	2384.6	1803.5	2806.1	2214.8
	Average	2004.9	2006.0	1998.3	22208.4	35956.3	2838.7	2204.6

Figure 6.24 presents the boxplots with the generalization errors achieved by all the methods in the 30 folds. In general the SLM variants present smaller performance variations across the different folds. Table 6.3 presents the results from the statistical comparisons between the SLM variants and the remaining methods. p-values are only presented for the comparisons with statistically significant differences. The comparisons where no statistically significant differences are found are left blank. The statistical superiority always refers to the SLM variant being superior to the other method in comparison. The reverse never occurs. In other words, none of the non-evolutionary supervised learning methods tested is superior to any SLM variant. In the 36 comparisons performed, the SLM variants achieve significantly superior outcomes in 25 comparisons. In the remaining 11 comparisons no significant differences are found.

Table 6.3: p-values of each comparison with statistically significant differences

Dataset	**SLM variant**	**Other method**			
		SVR	MLP	RT	LR
Bio	FLS EDV	1.8e-02	1.9e-05	4.1e-03	1.5e-02
	FLS TIE	2.3e-03	6.5e-06	1.5e-03	1.3e-03
	OLS EDV	1.2e-02	5.8e-05	7.1e-03	6.8e-03
PPB	FLS EDV			1.1e-02	6.8e-03
	FLS TIE		2.2e-02	2.3e-03	3.7e-04
	OLS EDV			2.8e-02	5.0e-03
LD50	FLS EDV	2.7e-02		6.7e-04	
	FLS TIE	2.8e-02		7.5e-04	
	OLS EDV	2.7e-02		4.6e-04	

Overall, the SLM variants achieved competitive outcomes. However, the experiments

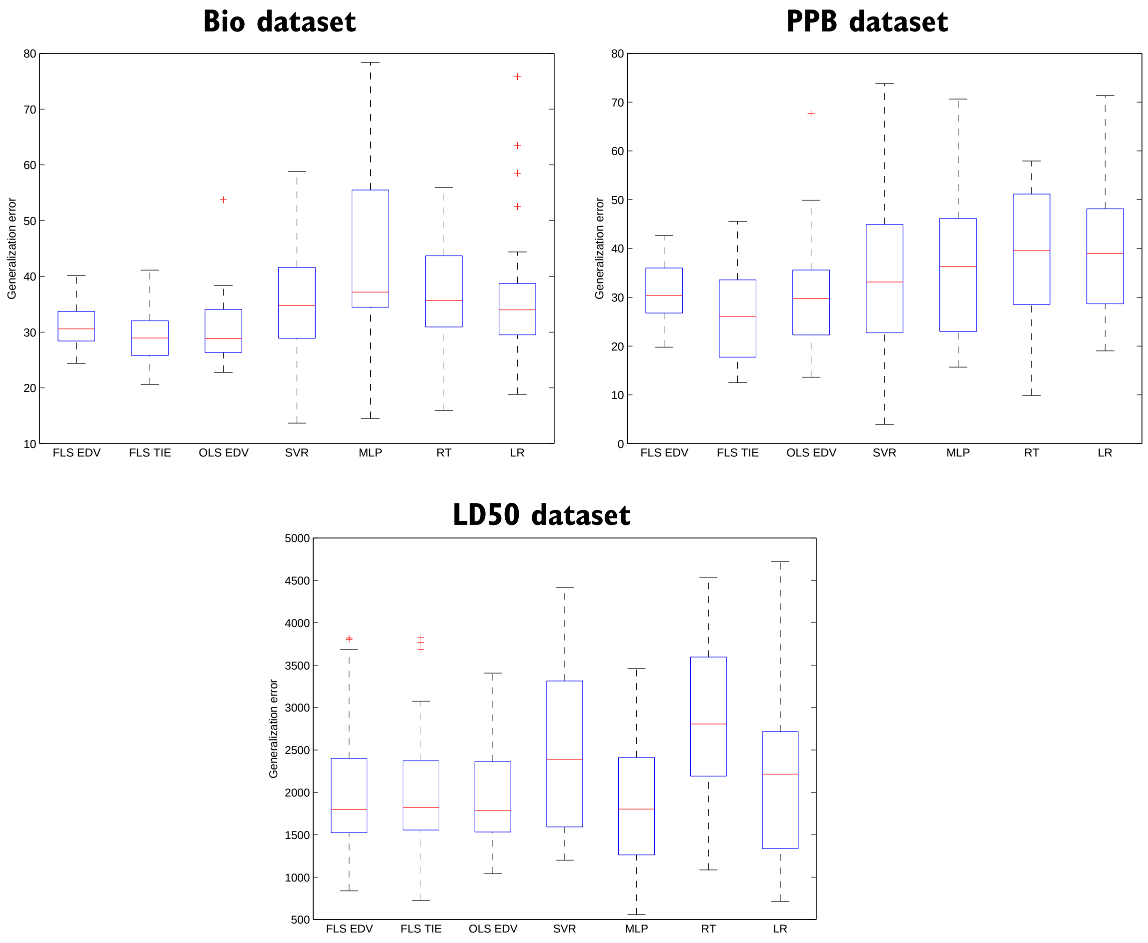


Figure 6.24: Boxplots with the generalization errors across the 30 folds

conducted in this section were exploratory in nature. Further empirical testing is needed to fully assess the general competitiveness of the SLM. Experiments need to be extended to other datasets, which should also include assessments in classification tasks. Further comparisons should also involve the optimization of the parameters of each method in each dataset (e.g., by using nested cross-validation). This is also valid for the SLM variants. The SLM was also tested in its simplest form, i.e., in the single hidden layer case. As the geometric semantic mutation was also defined for any number of hidden layers, the future SLM assessments should include the exploration of the effect of using several hidden layers.

A final note on the computational time of the SLM variants. Figure 6.25 presents the computational time in seconds of the SLM variants tested in this section. OLS EDV requires around 1 second in computational time in all datasets. The FLS variants take a little longer to stop, but are also very fast. In the Bio and PPB datasets, the FLS variants require around 4 or 5 seconds to compute. The computational times of the same variants in the LD50 dataset are a bit misleading. A bigger fixed learning step would reduce the computational time to values similar to the Bio and PPB datasets. As it was seen in figure 6.4, the usage of bigger learning step (100) in the LD50 dataset would be suitable if a stopping criterion was used. Overall, the SLM variants are demonstrably efficient in terms of computational time.

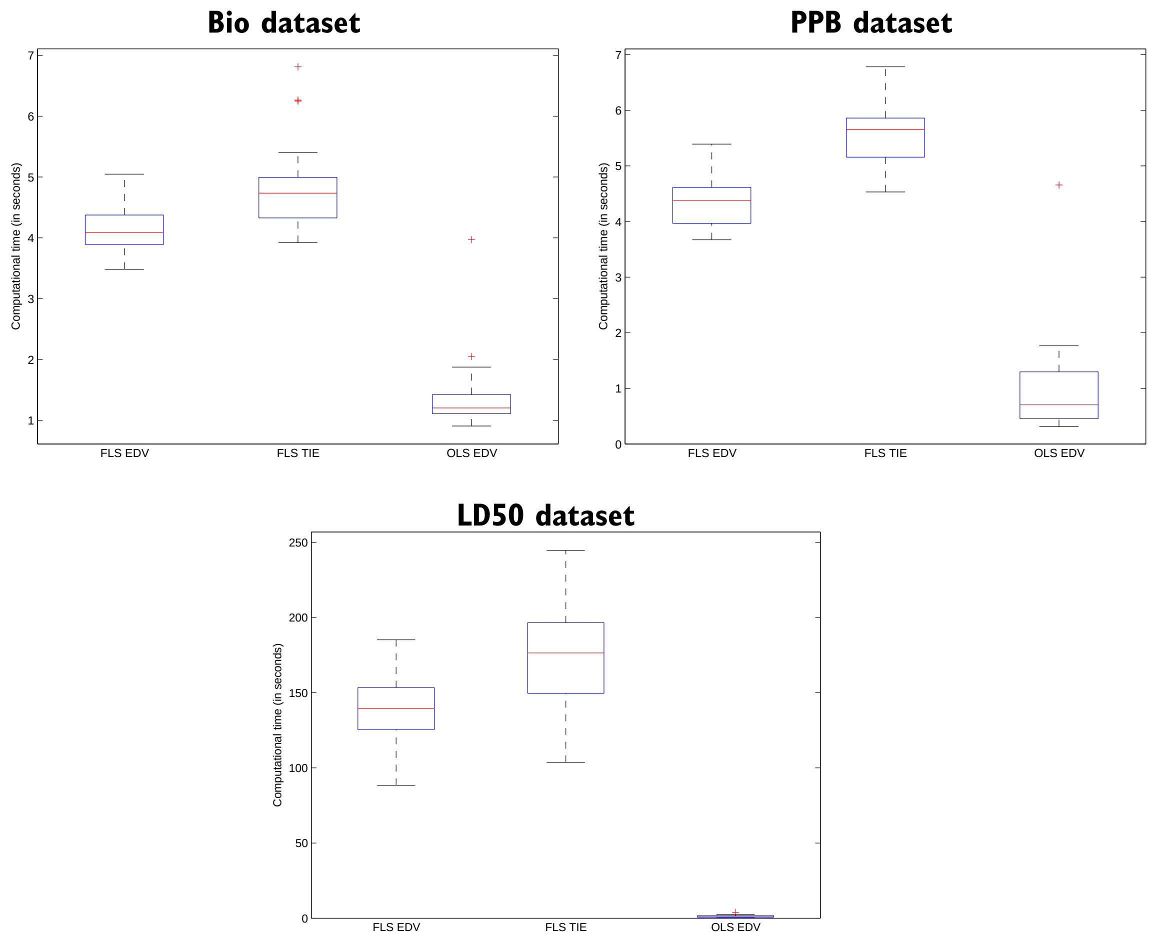


Figure 6.25: Computational time of each SLM variant

7 Conclusions

7.1 Summary

The results in this dissertation show that Standard GP is prone to overfitting. To address this issue, a set of approaches was studied in chapter 4. This set of approaches relies on a different and dynamic use of the available training data. The goal of the first approach was to study the generalization effect of using very small dynamic subsets of training data. The extreme case of using only a single training instance was also considered. This first approach works by, at each generation, defining the fitness of each individual as the performance on a randomly selected subset of the training data. This implies that only individuals that perform well on various different training data subsets are able to reproduce and/or remain in the population. The results showed that using very small dynamic subsets of training data is indeed effective at avoiding overfitting. Besides generalizing better than Standard GP, the resulting individuals were also considerably smaller. The best results were obtained either when using a single training instance or when using 5% of the total training data available. These results showed that the best surviving individuals have indeed captured the underlying relationships of the data instead of overfitting it. Given the particularly interesting effect of using a single dynamically changing training instance, other similar approaches were proposed. These rely on interleaving a single training instance with periodically using all of the training data. The rationale is based on trying to increase the training data learning rate, while still avoiding overfitting. The results show that it is possible to achieve these outcomes if a clear preference is given to using more generations with a single training instance. A normalized comparison was also provided that showed that the best approaches are not simply delaying overfitting by presenting less data to the search method. These best approaches are indeed avoiding overfitting. The approaches tested in chapter 4 showed that it is possible to avoid overfitting without imposing any constraint on the complexity of the individuals. As a positive side-effect, these approaches also contribute to faster GP runs as the fitness evaluation is performed using less training instances.

Chapter 5 provided an analysis of the generalization ability of GSGP. This analysis showed that the generalization outcome is considerably dependent on the implementation of the mutation operator. The original mutation operator is shown to be prone to overfitting, while a slightly modified mutation operator can effectively protect against overfitting. GSGP with the modified mutation operator can achieve a competitive generalization. A justification for these outcomes was provided by remarking some similarities with the area of Ensemble Learning. A novel geometric semantic mutation was also proposed and experimentally tested. This novel mutation can considerably improve the training data learning rate. It is also shown in the following chapter that, with an appropriate stopping criterion, the proposed mutation operator is able to achieve a competitive generalization while producing very small individuals. When using the mentioned stopping criterion, the evolutionary process only needs to run a few generations. Chapter 5 also showed how GSGP can be used to effectively find aligned individuals with arbitrary precision. This can be seen as an alternative way of conducting the evolutionary search, as the original supervised learning targets can be directly disregarded during the search process.

Chapter 6 showed how the GSGP mutation can be extended to NN. The most important consequence is that search can now be performed in the space of NN with the same unimodal error landscape as in GSGP. The proposed mutation operator excludes the need to use backpropagation to adjust the weights of the network. With the proposed mutation operator, a NN construction algorithm was proposed, named Semantic Learning Machine (SLM). It was empirically shown that with the SLM it is possible to perform an effective and efficient stochastic search in the space of NN. It is also interesting to point out that the SLM entails a free lunch, given that the error function induces a geometrical structure over the targets. This also applies to GSGP as previously stated. It was also shown that the mutation operator used within SLM allows for a simple way of controlling the sparseness at each mutation operation. This results in a structural simplification of the resulting NN. Two search stopping criteria were also proposed. Results showed that these stopping criteria were able to detect stopping points that result in a competitive generalization. The search was always stopped before overfitting started to occur. These stopping criteria were shown to be successful in SLM and GSGP. Given that the corresponding searches stopped early, the resulting NNs or individuals were also considerably smaller.

7.2 The Road Ahead

Experimentally assessing the SLM for the case of several hidden layers is one of the first future steps. It remains to be seen if the added hidden layers can provide a superior generalization or faster convergence, even though in terms of representation, the single hidden layer case is sufficient under the universal approximation results. It is also important to study if the effectiveness of the semantic stopping criteria remains unaltered in the case of several hidden layers. The geometric semantic mutation defined for feedforward NNs can also be extended for the case of recurrent NNs. This extension is possible under mild restrictions. The corresponding experimental assessment of the geometric semantic mutation for recurrent NNs is also an important future step.

As is commonly the case, an even more extended experimental assessment would be valuable. This extended experimental assessment should include: extending the regression analysis to other datasets; providing extended comparisons with other supervised learning methods; and providing results for classification tasks. Of particular interest is the application of the best performing methods to other challenging tasks.

As a field, GP has been gradually overcoming the methodological flaws that were prevalent in the early years, and the topics of generalization and overfitting have been receiving considerably more focus. Addressing these topics is fundamental in order for GP to be more widely recognized as a valuable technique within the larger supervised learning area. The recent developments in terms of more efficient search operators, and the recent novel approaches focusing on generalization and overfitting, bring GP closer to a wider recognition.

Bibliography

(2009). Apophysis. `https://sourceforge.net/projects/apophysis/`.

Amil, N. M., Bredeche, N., Gagné, C., Gelly, S., Schoenauer, M., and Teytaud, O. (2009). A statistical learning perspective of genetic programming. In European Conference on Genetic Programming, pages 327--338. Springer.

Archetti, F., Lanzeni, S., Messina, E., and Vanneschi, L. (2007). Genetic programming for computational pharmacokinetics in drug discovery and development. Genetic Programming and Evolvable Machines, 8(4):413--432.

Banzhaf, W., Francone, F. D., and Nordin, P. (1996). The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets. In Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, PPSN IV, pages 300--309, London, UK. Springer-Verlag.

Becker, L. A. and Seshadri, M. (2003). Comprehensibility and overfitting avoidance in genetic programming for technical trading rules. Technical report, Worcester Polytechnic Institute.

Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies--a comprehensive introduction. Natural computing, 1(1):3--52.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory, pages 144--152. ACM.

Brameier, M. F. and Banzhaf, W. (2007). Linear genetic programming. Springer Science & Business Media.

Breiman, L. (1996a). Bagging predictors. Machine learning, 24(2):123--140.

Breiman, L. (1996b). Bias, variance, and arcing classifiers. Technical report, Statistics Department, University of California, Berkeley, CA, USA.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA.

Castelli, M., Manzoni, L., Silva, S., and Vanneschi, L. (2010). A comparison of the generalization ability of different genetic programming frameworks. In IEEE Congress on Evolutionary Computation, pages 1--8.

Castelli, M., Silva, S., and Vanneschi, L. (2014). A c++ framework for geometric semantic genetic programming. Genetic Programming and Evolvable Machines, 16(1):73--81.

Cavaretta, M. J. and Chellapilla, K. (1999). Data mining using genetic programming: The implications of parsimony on generalization error. In IEEE Congress on Evolutionary Computation, pages 1330--1337. IEEE Press.

Chan, K. Y., Kwong, C. K., Dillon, T. S., and Tsim, Y. C. (2011). Reducing overfitting in manufacturing process modeling using a backward elimination based genetic programming. Appl. Soft Comput., 11:1648--1656.

Chen, Q., Xue, B., Shang, L., and Zhang, M. (2016). Improving generalisation of genetic programming for symbolic regression with structural risk minimisation. In Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, pages 709--716. ACM.

Chen, S.-H. and Kuo, T.-W. (2003). Overfitting or poor learning: a critique of current financial applications of gp. In Proceedings of the 6th European conference on Genetic programming, EuroGP '03, pages 34--46, Berlin, Heidelberg. Springer-Verlag.

Costelloe, D. and Ryan, C. (2009). On improving generalisation in genetic programming. In European Conference on Genetic Programming, pages 61--72. Springer.

Cramer, N. L. (1985). A representation for the adaptive generation of simple sequential programs. In Proceedings of the First International Conference on Genetic Algorithms, pages 183--187.

Da Costa, L. E. and Landry, J.-A. (2006). Relaxed genetic programming. In Proceedings of the 8th annual conference on Genetic and evolutionary computation, GECCO '06, pages 937--938, New York, NY, USA. ACM.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation, 6(2):182--197.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Multiple classifier systems, pages 1--15. Springer.

Domingos, P. (1999). The role of occam's razor in knowledge discovery. Data Min. Knowl. Discov., 3:409--425.

Domingos, P. (2012). A few useful things to know about machine learning. Communications of the ACM, 55(10):78--87.

Ehrenfeucht, A., Haussler, D., Kearns, M., and Valiant, L. (1989). A general lower bound on the number of examples needed for learning. Information and Computation, 82(3):247--261.

Eiben, A. E. and Jelasity, M. (2002). A critical note on experimental research methodology in ec. In Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002), volume 58, pages 2--587.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). Artificial Intelligence Through Simulated Evolution. John Wiley & Sons.

Foreman, N. and Evett, M. (2005). Preventing overfitting in gp with canary functions. In Proceedings of the 2005 conference on Genetic and evolutionary computation, GECCO '05, pages 1779--1780, New York, NY, USA. ACM.

Forsyth, R. (1981). Beagle-a darwinian approach to pattern recognition. Kybernetes, 10(3):159--166.

Freund, Y. and Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In European conference on computational learning theory, pages 23--37. Springer.

Friedman, J. H. (1997). On bias, variance, 0/1—loss, and the curse-of-dimensionality. Data mining and knowledge discovery, 1(1):55--77.

Gagné, C., Schoenauer, M., Parizeau, M., and Tomassini, M. (2006). Genetic programming, validation sets, and parsimony pressure. In EuroGP '06, pages 109--120.

Gathercole, C. and Ross, P. (1994). Dynamic training subset selection for supervised learning in genetic programming. In In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Manner, editors, Parallel Problem Solving from Nature III, pages 312--321. Springer-Verlag.

Giacobini, M., Tomassini, M., and Vanneschi, L. (2002). Limiting the number of fitness cases in genetic programming using statistics. In PPSN '02, pages 371--380.

Gonçalves, I. and Silva, S. (2013). Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In Genetic Programming, pages 73--84. Springer.

Gonçalves, I., Silva, S., and Fonseca, C. M. (2015a). On the generalization ability of geometric semantic genetic programming. In Genetic Programming, pages 41--52. Springer.

Gonçalves, I., Silva, S., and Fonseca, C. M. (2015b). Semantic learning machine: A feed-forward neural network construction algorithm inspired by geometric semantic genetic programming. In Progress in Artificial Intelligence, volume 9273 of Lecture Notes in Computer Science, pages 280--285. Springer.

Gonçalves, I., Silva, S., Fonseca, C. M., and Castelli, M. (2016). Arbitrarily close alignments in the error space: A geometric semantic genetic programming approach. In Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, pages 99--100. ACM.

Gonçalves, I., Silva, S., Melo, J. B., and Carreiras, J. M. B. (2012). Random sampling technique for overfitting control in genetic programming. In Genetic Programming, pages 218--229. Springer.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. ACM SIGKDD explorations newsletter, 11(1):10--18.

Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence, 12(10):993--1001.

Holland, J. H. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359--366.

Iba, H. (1999). Bagging, boosting, and bloating in genetic programming. In Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2, pages 1053--1060. Morgan Kaufmann Publishers Inc.

Iba, H., Sato, T., and de Garis, H. (1994). System identification approach to genetic programming. In Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, pages 401--406. IEEE.

Keijzer, M. and Babovic, V. (2000). Genetic programming, ensemble methods and the bias/variance tradeoff--introductory investigations. In European Conference on Genetic Programming, pages 76--90. Springer.

Kennedy, T. (1997). Managing the drug discovery/development interface. Drug Discovery Today, 2(10):436--444.

Kola, I. and Landis, J. (2004). Can the pharmaceutical industry reduce attrition rates? Nature Reviews Drug Discovery, 3(8):711--716.

Kötzing, T., Neumann, F., and Spöhel, R. (2011). Pac learning and genetic programming. In Proceedings of the 13th annual conference on Genetic and evolutionary computation, pages 2091--2096. ACM.

Koza, J. R. (1989). Hierarchical genetic algorithms operating on populations of computer programs. In IJCAI, pages 768--774.

Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems). The MIT Press, 1 edition.

Koza, J. R. (2010). Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines, 11(3-4):251--284.

Kushchu, I. (2002). An evaluation of evolutionary generalisation in genetic programming. Artif. Intell. Rev., 18:3--14.

Langdon, W. B. (2000). Size fair and homologous tree crossovers for tree genetic programming. Genetic programming and evolvable machines, 1(1-2):95--119.

Lichman, M. (2013). UCI machine learning repository.

Liu, Y. and Khoshgoftaar, T. (2004). Reducing overfitting in genetic programming models for software quality classification. In Proceedings of the Eighth IEEE international conference on High assurance systems engineering, HASE '04, pages 56--65, Washington, DC, USA. IEEE Computer Society.

Luke, S. and Panait, L. (2002). Lexicographic parsimony pressure. In GECCO, volume 2, pages 829--836.

Mahler, S., Robilliard, D., and Fonlupt, C. (2005). Tarpeian bloat control and generalization accuracy. In EuroGP, pages 203--214.

Mambrini, A. and Oliveto, P. S. (2016). On the analysis of simple genetic programming for evolving boolean functions. In European Conference on Genetic Programming, pages 99--114. Springer.

McDermott, J., White, D. R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., et al. (2012). Genetic programming needs better benchmarks. In Proceedings of the 14th annual conference on Genetic and evolutionary computation, pages 791--798. ACM.

Miller, J. F. and Thomson, P. (2000). Cartesian genetic programming. In European Conference on Genetic Programming, pages 121--132. Springer.

Montana, D. J. (1995). Strongly typed genetic programming. Evolutionary computation, 3(2):199--230.

Montana, J. L., Alonso, C. L., Borges, C. E., and Crespo, J. L. (2009). Adaptation, performance and vapnik-chervonenkis dimension of straight line programs. In European Conference on Genetic Programming, pages 315--326. Springer.

Moody, J., Hanson, S., Krogh, A., and Hertz, J. A. (1995). A simple weight decay can improve generalization. Advances in neural information processing systems, 4:950--957.

Moraglio, A. (2007). Towards a Geometric Unification of Evolutionary Algorithms. PhD thesis, Department of Computer Science, University of Essex, UK.

Moraglio, A., Krawiec, K., and Johnson, C. G. (2012). Geometric semantic genetic programming. In Parallel Problem Solving from Nature-PPSN XII, pages 21--31. Springer.

Moraglio, A. and Mambrini, A. (2013). Runtime analysis of mutation-based geometric semantic genetic programming for basis functions regression. In Proceedings of the 15th annual conference on Genetic and evolutionary computation, pages 989--996. ACM.

Murphy, G. and Ryan, C. (2008a). Manipulation of convergence in evolutionary systems. In Genetic Programming Theory and Practice V, pages 33--52. Springer.

Murphy, G. and Ryan, C. (2008b). A simple powerful constraint for genetic programming. In European Conference on Genetic Programming, pages 146--157. Springer.

Nikolaev, N., de Menezes, L. M., and Iba, H. (2002). Overfitting avoidance in genetic programming of polynomials. In Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress - Volume 02, CEC '02, pages 1209--1214, Washington, DC, USA. IEEE Computer Society.

Nikolaev, N. Y. and Iba, H. (2001). Regularization approach to inductive genetic programming. IEEE-EC, 5:359--375.

Nordin, P. (1994). A compiling genetic programming system that directly manipulates the machine code. Advances in genetic programming, 1:311--331.

O'Neill, M., Vanneschi, L., Gustafson, S., and Banzhaf, W. (2010). Open issues in genetic programming. Genetic Programming and Evolvable Machines, 11:339--363.

Paris, G., Robilliard, D., and Fonlupt, C. (2003). Exploring overfitting in genetic programming. In Artificial Evolution, pages 267--277.

Pennachin, C., Looks, M., and de Vasconcelos, J. A. (2011). Improved time series prediction and symbolic regression with affine arithmetic. In Riolo, R., Vladislavleva, E., and Moore, J., editors, Genetic Programming Theory and Practice IX, Genetic and Evolutionary Computation. Springer, Ann Arbor, USA.

Poli, R. (2003). A simple but theoretically-motivated method to control bloat in genetic programming. In European Conference on Genetic Programming, pages 204--217. Springer.

Poli, R. and Graff, M. (2009). There is a free lunch for hyper-heuristics, genetic programming and computer scientists. In European Conference on Genetic Programming, pages 195--207. Springer.

Poli, R., Graff, M., and McPhee, N. F. (2009). Free lunches for function and program induction. In Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms, pages 183--194. ACM.

Poli, R., Vanneschi, L., Langdon, W. B., and McPhee, N. F. (2010). Theoretical results in genetic programming: the next ten years? Genetic Programming and Evolvable Machines, 11(3-4):285--320.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. Neural Netw., 11:761--767.

Quinlan, J. R. (1987). Simplifying decision trees. International Journal of Man-Machine Studies, 27(3):221--234.

Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann.

Rissanen, J. (1978). Modeling by shortest data description. Automatica, 14:465--471.

Robilliard, D. and Fonlupt, C. (2001). Backwarding : An overfitting control for genetic programming in a remote sensing application. In Artificial Evolution, pages 245--254.

Ruberto, S., Vanneschi, L., Castelli, M., and Silva, S. (2014). Esagp--a semantic gp framework based on alignment in the error space. In Genetic Programming, pages 150--161. Springer.

Ryan, C., Collins, J., and Neill, M. O. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In European Conference on Genetic Programming, pages 83--96. Springer.

Schaffer, C. (1994). A conservation law for generalization performance. In Proceedings of the 11th international conference on machine learning, pages 259--265.

Schapire, R. E. (1990). The strength of weak learnability. Machine learning, 5(2):197--227.

Schölkopf, B. and Smola, A. J. (2002). Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press.

Silva, S. and Costa, E. (2009). Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines, 10:141--179.

Silva, S. and Dignum, S. (2009). Extending operator equalisation: Fitness based self adaptive length distribution for bloat free gp. In European Conference on Genetic Programming, pages 159--170. Springer.

Silva, S., Ingalalli, V., Vinga, S., Carreiras, J. M., Melo, J. B., Castelli, M., Vanneschi, L., Gonçalves, I., and Caldas, J. (2013). Prediction of forest aboveground biomass: an exercise on avoiding overfitting. In European Conference on the Applications of Evolutionary Computation, pages 407--417. Springer.

Teller, A. and Andre, D. (1997). Automatically choosing the number of fitness cases: The rational allocation of trials. Genetic Programming, 97:321--328.

Tornay, S. C. (1938). Ockham: Studies and Selections, chapter Commentarium in Sententias, I, 27. Open Court Publishers, La Salle, IL.

Uy, N. Q., Hien, N. T., Hoai, N. X., and O'Neill, M. (2010). Improving the generalisation ability of genetic programming with semantic similarity based crossover. In EuroGP, pages 184--195.

Valiant, L. G. (1984). A theory of the learnable. Communications of the ACM, 27(11):1134--1142.

Vanneschi, L., Archetti, F., Castelli, M., and Giordani, I. (2009). Classification of oncologic data with genetic programming. J. Artif. Evol. App., 2009:6:1--6:13.

Vanneschi, L., Castelli, M., Manzoni, L., and Silva, S. (2013). A new implementation of geometric semantic gp and its application to problems in pharmacokinetics. In Genetic Programming, pages 205--216. Springer.

Vanneschi, L., Castelli, M., and Silva, S. (2010). Measuring bloat, overfitting and functional complexity in genetic programming. In Genetic and Evolutionary Computation Conference, pages 877--884.

Vanneschi, L. and Gustafson, S. (2009). Using crossover based similarity measure to improve genetic programming generalization ability. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, GECCO '09, pages 1139--1146, New York, NY, USA. ACM.

Vanneschi, L., Rochat, D., and Tomassini, M. (2007). Multi-optimization for generalization in symbolic regression using genetic programming. In Proceedings of the 2nd Annual Italian Workshop on Artificial Life and Evolutionary Computation (WIVACE'07).

Vanneschi, L. and Silva, S. (2009). Using operator equalisation for prediction of drug toxicity with genetic programming. In Proceedings of the 14th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence, EPIA '09, pages 65--76, Berlin, Heidelberg. Springer-Verlag.

Vapnik, V. N. (1995). The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA.

Vladislavleva, E. J., Smits, G. F., and Den Hertog, D. (2009). Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. Trans. Evol. Comp, 13:333--349.

Whigham, P. A. et al. (1995). Grammatically-based genetic programming. In Proceedings of the workshop on genetic programming: from theory to real-world applications, volume 16, pages 33--41.

White, D. R., Mcdermott, J., Castelli, M., Manzoni, L., Goldman, B. W., Kronberger, G., Jaśkowski, W., O'Reilly, U.-M., and Luke, S. (2013). Better gp benchmarks: community survey results and proposals. Genetic Programming and Evolvable Machines, 14(1):3--29.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. Neural computation, 8(7):1341--1390.

Wolpert, D. H. (2002). The supervised learning no-free-lunch theorems. In Soft Computing and Industry, pages 25--42. Springer.

Zhang, B.-T. and Mühlenbein, H. (1995). Balancing accuracy and parsimony in genetic programming. Evol. Comput., 3:17--38.

Zitzler, E., Laumanns, M., Thiele, L., et al. (2001). Spea2: Improving the strength pareto evolutionary algorithm. In Eurogen, volume 3242, pages 95--100.